Reducing The Need For Trusted Parties In Cryptography

Michel Abdalla

École normale supérieure
LIENS – École Doctorale de Sciences Mathématiques de Paris Centre

Reducing The Need For Trusted Parties In Cryptography

THÈSE D'HABILITATION

présentée pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches de l'École normale supérieure

(Spécialité Informatique)

par

Michel Ferreira Abdalla

Soutenue publiquement le 24 novembre 2011 devant le jury composé de

Jean-Sébastien Coron	*Rapporteur*
Antoine Joux	*Rapporteur*
Kenny Paterson	*Rapporteur*
Ronald Cramer	*Examinateur*
Marc Fischlin	*Examinateur*
Nigel Smart	*Examinateur*
Jacques Stern	*Examinateur*
David Pointcheval	*Directeur*

Travaux effectués au sein de l'Équipe de Cryptographie
du Laboratoire d'Informatique de l'École normale supérieure

Acknowledgements

First and foremost, I would like to thank David Pointcheval. I can honestly say that I would not be where I am today without his support and guidance, including the supervision of this thesis, and especially for his friendship. Had it not been for his enthusiasm for cryptography and persistence, I would probably still be in the U.S. or back in Brazil.

In addition to David, I would also like to thank Jean-Sébastien Coron, Ronald Cramer, Marc Fischlin, Antoine Joux, Kenny Paterson, Nigel Smart, and Jacques Stern for accepting to be part of this jury. Special thanks to Antoine, Jean-Sébastien, and Kenny, for also accepting to be *Rapporteurs* for this thesis.

Next, I would like to thank the following, both current and former, colleagues from the ENS Crypto team: Charles Bouillaguet, Bruno Blanchet, Olivier Blazy, David Cadé, Dario Catalano, Yuanmi Chen, Céline Chevalier, Patrick Derbez, Léo Ducas, Dario Fiore, Pierre-Alain Fouque, Georg Fuchsbauer, Malika Izabachène, Jérémy Jean, Roch Lescuyer, Vadim Lyubashevsky, David Naccache, Gregory Neven, Phong Nguyen, Miriam Paiola, Duong Hieu Phan, David Pointcheval, Oded Regev, Jacques Stern, Mario Strefler, Mehdi Tibouchi, and Damien Vergnaud. Their friendship and shared passion for numbers has led to numerous discussions, papers, and most importantly has created an enjoyable work environment. I would also like to thank the administrative and technical staff of the Computer Science department of ENS for their everyday support: Michelle Angely, Jacques Beigbeder, Lise-Marie Bivard, Isabelle Delais, Joëlle Isnard, Jean Claude Lovato, Valérie Mongiat, and Ludovic Ricardou.

The work on which this thesis is based was made possible by numerous collaborations, and I would like to thank all of those with whom I have had the opportunity to work. In particular, I would like to give a special acknowledgement to my most frequent collaborators: Mihir Bellare, Emmanuel Bresson, Dario Catalano, Céline Chevalier, Olivier Chevassut, Alex Dent, Pierre-Alain Fouque, Eike Kiltz, John Malone-Lee, Chanathip Namprempre, Gregory Neven, David Pointcheval, and Nigel Smart.

I hold a special place in my acknowledgements for Mihir Bellare and Phil Rogaway, who got me started in Cryptography. Thanks to them, I published my first paper in the field, and I have since been hooked.

Upon my arrival in France, I was fortunate enough to have Marcelo "Baiano" as a dear friend. Not only did he help me get settled in Paris, but he also showed me the "French" ropes. I am also extremely lucky to have met Dario Catalano, who is the one person most responsible for jump-starting my social life in Paris. Without his cherished friendship and sense of humor, my wife and I would have driven each other up the wall.

Finally, I would like to thank my wife Stacie for her love and support. Not only has she been able to put up with the crazy hours and deadlines throughout the years, but she has even learned to speak the crypto lingo. I also thank her for having the courage to move with me into a foreign country whose language she could barely speak and for accepting to "survive" in a 20 m^2 apartment with me and our cat Tigger.

Abstract

Trusted parties are fundamental for the establishment of secure communication among users. Such is the case, for example, when establishing a trusted relationship between users and certain public information in a public-key infrastructure for public-key encryption and signature schemes or when storing high-entropy secret keys in a cryptographic device. Clearly, if the trusted party misbehaves in either of these situations, then the overall security of the scheme or protocol in which we are interested can be adversely affected.

There are several ways in which one can try to reduce the amount of trust in third parties, such as making the task of recovering the secret key harder for the adversary, as in distributed cryptosystems or minimizing the damage caused by secret-key exposures, as in forward-secure and intrusion-resilient cryptosystems. In this thesis, we consider two additional methods.

The first one, which goes by the name of password-based key exchange, is to assume that the secret keys used in authenticated key exchange protocols have low entropy and do not need to be stored in a cryptographic device. In spite of the low entropy of secret keys, such protocols can still provide a level of assurance which may be sufficient for most applications.

The second method for reducing the amount of trust in third parties is to use an identity-based cryptosystem, in which the public key of a user can be an arbitrary string such as an email address. As identity-based cryptosystems provide collusion resistance, they can also be used to lessen the damage caused by secret-key exposures by generating independent secret keys for different time periods or devices. Moreover, identity-based cryptosystems can allow users to have a more fine-grained control over the decryption capabilities of third parties, further limiting the harmful consequences due to their misbehavior.

Keywords: Provable security, password-based cryptography, identity-based cryptography.

Résumé

Les tiers de confiance sont essentiels aux communications sécurisées. Par exemple, dans une infrastructure de gestion de clés, l'autorité de certification est la clé de voute de l'authentification en garantissant le lien entre une identité et une clé publique. Une carte à puce se doit, pour sa part, d'assurer la confidentialité et l'intégrité des données secrètes lorsqu'elle sert de stockage de données cryptographiques. En effet, si ces garanties sont mises en défaut dans l'une de ces situations, alors la sécurité globale du système peut en être affectée.

Plusieurs approches permettent de réduire l'importance des tiers de confiance, telles qu'accroître la difficulté de recouvrer la clé secrète, en la distribuant parmi plusieurs entités, ou limiter l'impact d'une fuite d'information secrète, comme dans les cryptosystèmes « intrusion-resilient » ou « forward-secure ». Dans cette thèse, nous considérons deux méthodes complémentaires.

La première méthode consiste à utiliser des mots de passe, ou des clés secrètes de faible entropie, qui n'ont pas besoin d'être stockés dans un dispositif cryptographique sécurisé. Malgré la faible entropie du secret, de tels protocoles peuvent fournir un niveau d'assurance satisfaisant pour la plupart des applications. On considère en particulier la mise en accord de clés.

La deuxième méthode limite le besoin de garantie de la part des tiers de confiance en utilisant un cryptosystème basé sur l'identité, dans lequel la clé publique d'un utilisateur peut être une chaîne de caractères arbitraire, telle qu'une adresse email. Comme ces systèmes fournissent une résistance aux collusions, ils peuvent aussi être utilisés pour réduire les dommages causés par l'exposition de clés secrètes en générant des secrets indépendants pour chaque période de temps ou pour chaque périphérique/entité. Par ailleurs, ces systèmes basés sur l'identité permettent aux utilisateurs d'avoir un contrôle plus fin sur les capacités de déchiffrement des tiers, en limitant les conséquences liées à un mauvais usage.

Mots-clés : Sécurité prouvée, cryptographie basée sur des mots de passe, cryptographie basée sur l'identité.

Contents

Part I

Reducing The Need For Trusted Parties In Cryptography

Introduction

The primary goal of cryptography is to enable parties to communicate securely over an insecure channel, which may be under the control of an adversary. Though originally used mainly for the purpose of protecting the privacy of messages, cryptography now encompasses many other goals, such as guaranteeing the integrity of messages being exchanged or the authenticity of the sender.

For most of its history, cryptography was essentially a game played between designers and attackers in which one side would try to outsmart the other by conceiving ad hoc attack and defense mechanisms for their particular goals [Bel11]. Although this ad hoc aspect may always be present in the field, cryptography has since become a well established science, with clear security definitions and objectives.

The exact security objectives in which one might be interested will be determined by the application one has in mind and may depend on many factors, such as how powerful adversaries may be or the type of information that needs to be protected. The two most common goals usually considered are data privacy and authenticity. While the goal of *data privacy* is to keep unintended parties from learning the contents of the message being sent over the channel, *data authenticity* aims at guaranteeing that the contents of the message have not been tampered with during the course of transmission.

Encryption schemes. The usual way to achieve data privacy and allow parties to exchange messages privately over an insecure channel is to use an *encryption* scheme. In these schemes, there are two types of keys, known as the encryption and decryption keys. Whenever a sender wants to send a private message, or plaintext, to a receiver, he first converts it into an enciphered form, called a ciphertext, with the help of the encryption key and then sends the latter to the receiver. Upon the receipt of a ciphertext, the receiver can use the decryption key to recover the original plaintext. The algorithms used to generate the ciphertext and to recover the plaintext are known as the encryption and decryption algorithms respectively.

The classical example of an encryption scheme is Shannon's one-time pad [Sha49]. In the one-time pad scheme, both the sender and receiver have to share a secret key whose length is at least that of the message being exchanged. To encrypt a message, the sender simply computes the bit-wise XOR of the message with the shared secret key, which serves as the encryption key, and sends it to the receiver. Upon receiving a ciphertext, the receiver can recover the original message in a similar manner using the shared secret key as the decryption key. Despite its simplicity, Shannon showed that the one-time pad is actually as secure as it gets since, as long as the shared secret key does not get reused, it is impossible for an adversary to gain any information about the plaintext, other than its length, from the ciphertext. Moreover, this is the case even if the adversary has unlimited computational resources. Its main drawback lies in the key size, which needs to be at least as long as the message, thus imposing a limit on the amount of information that can be securely transmitted.

Signature schemes. The standard way of achieving data authenticity is to use a *signature*

scheme. As in an encryption scheme, there are two types of keys, to which we refer as signing and verification keys. Using the signing key, a sender can create a signature for each message whose authenticity needs to be guaranteed. This signature will be attached to the message before its transmission and will serve as a testament to the fact that the message is authentic. After receiving a message together with its signature, the receiver can verify its authenticity by checking the validity of the signature with the help of the verification key. The algorithms used to generate the signature and to verify its validity are known as the signing and verification algorithms respectively.

Symmetric and asymmetric settings. In order for an encryption scheme to guarantee data privacy, it is clear that the decryption key used by the receiver should be kept hidden from the adversary or else the adversary could use it to recover the plaintext associated with any ciphertext sent over the channel. Likewise, to guarantee data authenticity, the signing key of a signature scheme needs to remain secret or else the adversary could fabricate authenticators for any message of its choice.

As for the encryption and verification keys used in these schemes, they might be either secret or not depending on the particular setting in which we are interested. In the *private-key* cryptographic setting, both encryption and verification keys are assumed to be unknown to the adversary. Due to the secrecy of these keys, both senders and receivers have to agree on their values before they can start communicating securely. Moreover, since these keys usually have high entropy, one may need a cryptographic device to safely store them. In this setting, signature schemes are also known as message authentication codes. On the other hand, in the *public-key* cryptographic setting, encryption and verification keys are not necessarily hidden from the adversary. They are only mathematically related to the corresponding decryption and signing keys and may be made public. Because the encryption and verification keys are usually equal to the decryption and signing keys, respectively, in the private-key setting and different from them in the public-key setting, we also refer to these settings as *symmetric* and *asymmetric*.

The notion of public-key cryptography was introduced by Diffie and Hellman [DH76] and is one of the main innovations of modern cryptography. Unlike private-key cryptography, public-key cryptography does not require previously established secrets to enable secure communication between parties. In public-key cryptography, each user has a pair of *public* and *secret* keys associated to him. While the public key is known to all parties, including the adversary, the secret key should only be known to the user with whom it is associated.

Since public and secret keys are mathematically related, it is always possible for an adversary to compute a corresponding secret key when given a public key. Hence, public-key cryptography should only be considered in an environment in which adversaries are assumed to be computationally bounded. In this *computational-complexity* setting, rather than *impossibility*, we talk about the *infeasibility* of breaking the security of a scheme.

In order to link users to their public keys, public-key encryption and signature schemes have to rely on the existence of a public-key infrastructure (PKI), where a trusted authority certifies the relation between users and their public keys by means of a digital signature. Since users need to know the public key associated with the trusted authority itself to be able to verify the validity of signatures issued by this authority, the latter public key needs to be pre-distributed to all users in the system.

Key exchange. Although public-key encryption and signature schemes can guarantee the privacy and authenticity of exchanged messages when users have access to certified copies of each other's public keys, their costs may be too high for certain applications. To avoid the use of such primitives and improve the overall efficiency, users often prefer to secure their communication via symmetric encryption and signature schemes. Unfortunately, this is only possible when users

are in possession of pre-established common secrets. To alleviate the need for pre-established secrets without significantly impacting the overall efficiency, another way of dealing with this problem is for users to first establish a common secret key via a key exchange protocol and to use this key to derive keys for symmetric encryption and signature schemes.

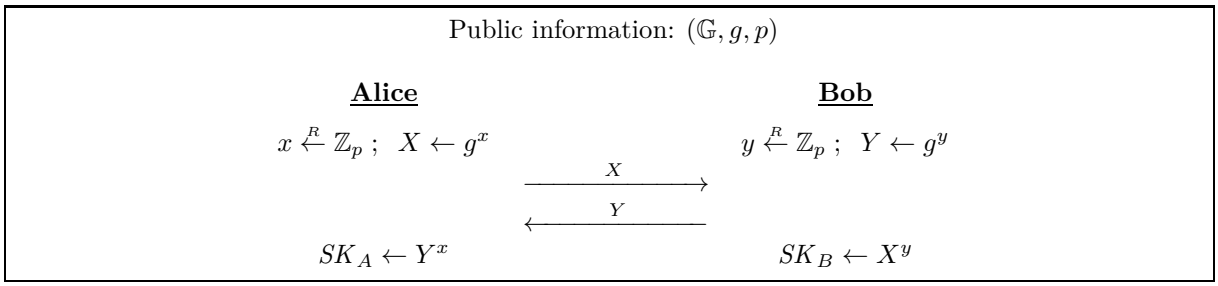


Figure 1: The Diffie-Hellman key exchange protocol [DH76]. The protocol works over a finite cyclic group $\mathbb{G}$ of prime order p, generated by an element g.

The classical example of a key exchange protocol is the Diffie-Hellman key exchange [DH76], which is depicted in Figure 1. Let $\mathbb{G}$ be a cyclic group of prime order p, let g be a generator for $\mathbb{G}$, and let Alice and Bob be two users willing to establish a shared secret key. In the Diffie-Hellman key exchange protocol, Alice chooses a secret value x uniformly at random from $\mathbb{Z}_p$ and then sends the ephemeral public key value $X = g^x$ to Bob. Likewise, Bob chooses a secret value y in $\mathbb{Z}_p$ and sends $Y = g^y$ to Alice. Finally, after receiving the values X and Y, both Alice and Bob can compute a common secret $SK = g^{xy}$ using their corresponding secret values.

Despite its simplicity, the Diffie-Hellman key exchange protocol can be shown to be secure under reasonable assumptions with respect to adversaries which only eavesdrop on the communication without altering it. However, the same is not true with respect to active adversaries. The problem with it is that the Diffie-Hellman key exchange protocol does not include any form of authentication and can be easily compromised by an adversary that plays the role of one of the parties in the protocol. It thus becomes clear that to avoid such impersonation attacks and achieve security even in the presence of active adversaries, both parties need to somehow authenticate each other in addition to establishing a common secret.

Authenticated key exchange. There are several ways in which one can embed authentication into a key exchange protocol. The first one, to which we refer as the 2-party asymmetric model, is to assume that each user in the system is associated with a pair of public and secret keys and to use public-key signature schemes to authenticate the communication between the parties. An example of a popular protocol in this model is the 2-party SIGMA protocol [Kra03] used as the basis for the signature-based modes of the Internet Key Exchange (IKE) protocol. It is worth noting that we implicitly assume the existence of a PKI when adopting this model.

A second method of adding authentication, known as the 3-party symmetric model, is to assume the existence of a trusted server with which all parties share a secret. In this model, users can use the help of the trusted server with which they share a key together with symmetric encryption and signature schemes to avoid active attacks during the key establishment protocol. A well-known protocol in this model is the 3-party *Kerberos* authentication system [SNS88]. The problem with this approach is that the trusted server needs to be present during the key establishment and this may possibly cause a bottleneck.

Finally, yet another form of obtaining an authenticated key exchange, referred to as the 2-party symmetric model, is to assume that all users in the system already share a common secret. As in 3-party model, users can use their pre-established secrets to secure their communication when establishing new common secrets.

The need for trusted parties. As the above discussion shows, trusted parties are fundamental for the establishment of secure communication among parties. Such is the case, for example, when establishing a trusted relationship between users and certain public information in a public-key infrastructure (PKI) for public-key encryption and signature schemes or when storing high-entropy secret keys in a cryptographic device. Clearly, if the trusted party misbehaves in either of these situations, then the overall security of the scheme or protocol in which we are interested can be adversely affected. For instance, if the trusted certificate authority of a PKI is compromised and issues certificates for users without verifying their identities, then the privacy or authenticity of the communication can no longer be guaranteed. Likewise, if the cryptographic device in which secret keys are stored gets compromised, then the information advantage that honest users have over the adversary in the form of a secret key may no longer exist.

There are several ways in which one can try to reduce the amount of trust in third parties. One method is to use distributed cryptosystems such as the one by Desmedt and Franklin in [DF92]. In their scheme, secret signing keys are shared among several servers so that the adversary needs to compromise a sufficiently large set of servers in order to recover a signing key. Another method of reducing the trust in third parties is to use a intrusion-resilient cryptosystem, such as the constructions by Dodis *et al.* in [DKXY02, DFK$^+$03], which tries to minimize the damage caused by secret-key exposures. A third method requiring fewer interactions among parties is to use forward-secure cryptosystems, as in the signature scheme of Bellare and Miner in [BM99]. In these schemes, even though the public key is fixed, the secret key is updated at regular intervals so that the compromise of the current secret key does not compromise the security of previous time periods.

Contributions. In this thesis, we consider two additional methods for reducing the amount of trust in third parties. The first one, which is described in Chapter 2, is to assume that the secret keys used in authenticated key exchange protocols have low entropy and do not need to be stored in a cryptographic device. As we show in Chapter 2, despite the low entropy of secret keys, such protocols can still provide a level of assurance which may be sufficient for most applications. The second method, which is described in Chapter 3, is to use a generalization of public-key cryptography known as identity-based cryptography, in which the public key of a user can be an arbitrary string such as his own identity or email address. As shown in [BF03], in addition to facilitating the deployment of a public key infrastructure, identity-based cryptosystems can have several other applications. Like intrusion-resilient cryptosystems, identity-based schemes can be used to minimize the damage caused by secret-key exposures by generating independent secret keys for different time periods or devices. Additionally, identity-based encryption schemes can also allow the sender to have a more fine-grained control over the decryption capabilities of the receiver with the appropriate use of attributes.

Outline. Since the schemes considered in this thesis follow the provable-security paradigm, we start by describing this paradigm in more detail in Chapter 1. Next, in Chapters 2 and 3, we present an overview of password-based and identity-based cryptography along with our main contributions to these areas. The main articles in which these contributions are based can be found in the appendix. Finally, we conclude this thesis by presenting some closing remarks and perspectives.

Chapter 1

Provable security

1.1 Introduction

Even though we have already mentioned in the previous chapter several properties that a cryptographic scheme may possess, such as data privacy and authenticity, we have not actually shown how one can prove that a cryptographic scheme has these properties. In the past, the most common approach to validate the security of a cryptographic scheme was to search for attacks and to declare a scheme secure if no attack is found that contradicts its security. Unfortunately, the problem with this approach is that we can never be certain that an attack does not exist. Hence, the security of the scheme can only be considered heuristic at best as the possibility that an attack exists cannot be excluded.

Another approach for proving the security of a cryptographic scheme, known as *provable security*, is to relate the security of a cryptographic scheme with that of its underlying primitives or computational problems. To achieve this goal, one needs to first specify the attacker's capabilities and the security goals that a given cryptographic scheme must meet. Next, one needs to specify the attacker's capabilities and the security goals for the underlying primitives and computational problems. Finally, one needs to provide a reduction which shows how to transform an adversary that breaks the security goals of a cryptographic scheme into an adversary against the security goals of the cryptographic primitives and computational problems on which the scheme is based.

One direct consequence of provable security is that it obviates the need to search for specific attacks against a cryptographic scheme. This is because as long as we are ready to believe that the underlying primitives are secure or the computational problems are hard, then there can be no attacks against the cryptographic scheme in question.

In order to illustrate how the provable security methodology can be used in practice, we also provide a concrete example on how to prove the security of a cryptographic scheme by considering the classical ElGamal public-key encryption scheme [ElG85a]. But before doing so, we start by recalling some of the standard tools and computational problems that are used in this thesis.

1.2 Basic tools

In this section, we recall some of the definitions and basic tools that will be used in the remaining chapters, such as the syntax of code-based games and the description of bilinear maps.

1.2.1 Notation and conventions

Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$, then $\{0,1\}^n$ denotes the set of n-bit strings, and $\{0,1\}^*$ is the set of all bit strings. The empty string is denoted ε. More generally, if S is a set, then S^n is the set of n-tuples of elements of S, $S^{\leq n}$ is the set of tuples of length at most n. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. If S is finite, then $x \xleftarrow{R} S$ denotes the assignment to x of an element chosen uniformly at random from S. If $\mathcal{A}$ is an algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ on input x, and if $\mathcal{A}$ is randomized, then $y \xleftarrow{R} \mathcal{A}(x)$ denotes that the output of an execution of $\mathcal{A}(x)$ with fresh coins assigned to y. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for polynomial time and "PTA" for polynomial-time algorithm or adversary. We denote by $k \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for every $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) \leq k^{-c}$ for all $k > k_c$, and it is said to be *overwhelming* if the function $|1 - \nu(k)|$ is negligible.

1.2.2 Code-based games

Some of the definitions in this thesis use code-based game-playing [BR06]. In such games, there exist procedures for initialization (**Initialize**) and finalization (**Finalize**) and procedures to respond to adversary oracle queries. A game G is executed with an adversary $\mathcal{A}$ as follows. First, **Initialize** executes and its outputs are the inputs to $\mathcal{A}$. Then $\mathcal{A}$ executes, its oracle queries being answered by the corresponding procedures of G. When $\mathcal{A}$ terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted $\mathrm{G}(\mathcal{A})$, is called the output of the game, and "$\mathrm{G}(\mathcal{A}) = y$" denotes the event that the output takes a value y. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true.

1.2.3 Represented groups

Let $\mathbb{G} = \langle g \rangle$ be a finite cyclic group of prime order p generated by an element g, where $k = |p|$ is the security parameter. Throughout this thesis, we will use the multiplicative notation for the group operation. Hence, g^0 denotes the identity element of $\mathbb{G}$ and g^u denotes the group element of $\mathbb{G}$ that results from multiplying u copies of g, for $u \in \mathbb{N}$. Note that $g^u = g^{u \bmod |\mathbb{G}|}$ by Lagrange's theorem.

Algorithms which operate on $\mathbb{G}$ will be given string representations of elements in $\mathbb{G}$. For that, we require an injective map $\_\_ : \mathbb{G} \to \{0,1\}^\ell$ associated to $\mathbb{G}$, where ℓ is the length of the representation of group elements. Similarly, when a number $i \in \mathbb{N}$ is an input to, or output of, an algorithm, it must be appropriately encoded, say in binary. We assume all necessary encoding methods are fixed, and do not normally write the $\_\_$ operators.

The schemes considered in this thesis are parameterized by a *group generator*, which is a PTA $\mathcal{G}$ that on input 1^k returns the description of a multiplicative group $\mathbb{G}$ of prime order p, where $2^k < p < 2^{k+1}$.

1.2.4 Bilinear maps

The pairing-based schemes that we consider in this thesis are parameterized by a *pairing parameter generator*, which is a PTA $\mathcal{G}$ that on input 1^k returns the description of two multiplicative groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $2^k < p < 2^{k+1}$. By admissible, we mean that the map is bilinear, non-degenerate, and efficiently computable. Bilinearity means that for all $a, b \in \mathbb{Z}_p^*$ and all $g \in \mathbb{G}$, we have $\hat{e}(g^a, g^b) = \hat{e}(g,g)^{ab}$. By non-degenerate, we mean that $\hat{e}(g,g) = 1$ if and only if $g = 1$.

1.3 Standard complexity assumptions

CDH and DDH Problems. Two of the most common computational problems in finite cyclic groups are the computational Diffie-Hellman (CDH) and the decisional Diffie-Hellman (DDH) problems. In the CDH problem, the adversary is given a tuple (g, g^a, g^b) for random integers $a, b \in \mathbb{Z}_p^*$ and a random generator $g \xleftarrow{R} \mathbb{G}^*$ and its goal is to compute g^{ab}. In the DDH problem, the goal is to distinguish (g, g^a, g^b, g^{ab}) from (g, g^a, g^b, g^c) when g is a random generator for $\mathbb{G}$ and a, b, c are chosen uniformly at random from $\mathbb{Z}_p^*$.

To define more precisely the CDH problem with respect to a group generator $\mathcal{G}$, we consider the game $\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$ described in Figure 1.1 using the notation of code-based games. The game is defined by two procedures and is executed with an adversary $\mathcal{A}$ as follows. The procedure **Initialize** chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$ and two random integers $x, y \xleftarrow{R} \mathbb{Z}_p^*$, computes $X \leftarrow g^x$ and $Y \leftarrow g^y$, and returns $((\mathbb{G}, p), g, X, Y)$ to $\mathcal{A}$. Eventually, the adversary ends the game by querying the **Finalize** procedure with a group element Z, which outputs true if $Z = g^{xy}$ and false, otherwise. The advantage $\mathbf{Adv}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the CDH problem relative to $\mathcal{G}$ is then defined as the probability that $\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$ outputs true. In other words,

$$\mathbf{Adv}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right].$$

In order to define more formally the DDH problem relative to a group generator $\mathcal{G}$, consider the games $\mathbf{Exp}^{\text{ddh-}\beta}_{\mathcal{G},k}(\mathcal{A})$ described in Figure 1.1 for $\beta \in \{0, 1\}$. Both games are defined by two procedures, which are executed with an adversary $\mathcal{A}$ as follows. In the game $\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{A})$, the procedure **Initialize** chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$ and random integers $x, y \xleftarrow{R} \mathbb{Z}_p^*$, sets $z \leftarrow xy \mod p$, computes $X \leftarrow g^x$, $Y \leftarrow g^y$, and $Z \leftarrow g^z$, and returns $((\mathbb{G}, p), g, X, Y, Z)$ to $\mathcal{A}$. In the game $\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{A})$, the procedure **Initialize** chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$ and random integers $x, y, z \xleftarrow{R} \mathbb{Z}_p^*$, computes $X \leftarrow g^x$, $Y \leftarrow g^y$, and $Z \leftarrow g^z$, and returns $((\mathbb{G}, p), g, X, Y, Z)$ to $\mathcal{A}$. In both games, the adversary eventually ends the game by querying the **Finalize** procedure with a guess β, which in turn returns true if $\beta' = 1$ and false, otherwise. The advantage $\mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the DDH problem relative to $\mathcal{G}$ is then defined as the probability that $\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{A})$ outputs true minus the probability that $\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{A})$ outputs true. That is,

$$\mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{A}) = \Pr\left[\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{A}) = \mathsf{true}\right].$$

Finally, the CDH and DDH problems are said to be hard relative to $\mathcal{G}$ if $\mathbf{Adv}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$ and $\mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{A})$ are negligible functions in k for all PTAs $\mathcal{A}$.

BDH and BDDH Problems. In the bilinear-map setting, two of the most common computational problems are the bilinear Diffie-Hellman (BDH) problem and its decisional version, the bilinear decisional Diffie-Hellman (BDDH) problem [BF01, Jou04]. While in the BDH problem, the goal is to compute $\hat{e}(g, g)^{abc}$ given a tuple (g, g^a, g^b, g^c) for random integers $a, b, c \in \mathbb{Z}_p^*$, in the BDDH problem, the goal is to distinguish the element $\hat{e}(g, g)^{abc}$ from a random element of $\mathbb{G}_T^*$.

More precisely, the advantage $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the BDH problem relative to a pairing parameter generator $\mathcal{G}$ is defined as the probability that the adversary $\mathcal{A}$ outputs $\hat{e}(g, g)^{abc}$ on input $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c)$ for randomly chosen $g \xleftarrow{R} \mathbb{G}^*$ and $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$. Using the code-based notation, $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$ corresponds to probability that the game $\mathbf{Exp}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$ returns true in Figure 1.2.

To define the advantage $\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},k}(\mathcal{A})$ of an adversary $\mathcal{A}$ in solving the BDDH problem relative to $\mathcal{G}$, we consider the game $\mathbf{Exp}^{\text{bddh-}\beta}_{\mathcal{G},k}(\mathcal{A})$ between $\mathcal{A}$ and a challenger in Figure 1.2

Game $\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$	**Game** $\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{A})$	**Game** $\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{A})$
proc Initialize(k)	**proc Initialize**(k)	**proc Initialize**(k)
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$	$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$	$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, p)$
$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$
$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$	$y \xleftarrow{R} \mathbb{Z}_p^*$; $Y \leftarrow g^y$
Return $(\overrightarrow{\mathbb{G}}, g, X, Y)$	$z \leftarrow ab \mod p$; $Z \leftarrow g^z$	$z \xleftarrow{R} \mathbb{Z}_p^*$; $Z \leftarrow g^z$
	Return $(\overrightarrow{\mathbb{G}}, g, X, Y, Z)$	Return $(\overrightarrow{\mathbb{G}}, g, X, Y, Z)$
proc Finalize(Z)	**proc Finalize**(β')	**proc Finalize**(β')
Return $(Z = g^{xy})$	Return $(\beta' = 1)$	Return $(\beta' = 1)$

Figure 1.1: Games $\mathbf{Exp}^{\text{cdh}}_{\mathcal{G},k}(\mathcal{A})$, $\mathbf{Exp}^{\text{ddh-0}}_{\mathcal{G},k}(\mathcal{A})$, and $\mathbf{Exp}^{\text{ddh-1}}_{\mathcal{G},k}(\mathcal{A})$ defining the advantage of an adversary $\mathcal{A}$ against the CDH and DDH problems relative to a group generator $\mathcal{G}$ and security parameter k.

Game $\mathbf{Exp}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$	**Game** $\mathbf{Exp}^{\text{bddh-0}}_{\mathcal{G},k}(\mathcal{A})$	**Game** $\mathbf{Exp}^{\text{bddh-1}}_{\mathcal{G},k}(\mathcal{A})$
proc Initialize(k)	**proc Initialize**(k)	**proc Initialize**(k)
$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \xleftarrow{R} \mathcal{G}(1^k)$
$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$	$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$	$\overrightarrow{\mathbb{G}} \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e})$
$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$
$a \xleftarrow{R} \mathbb{Z}_p^*$; $A \leftarrow g^a$	$a \xleftarrow{R} \mathbb{Z}_p^*$; $A \leftarrow g^a$	$a \xleftarrow{R} \mathbb{Z}_p^*$; $A \leftarrow g^a$
$b \xleftarrow{R} \mathbb{Z}_p^*$; $B \leftarrow g^b$	$b \xleftarrow{R} \mathbb{Z}_p^*$; $B \leftarrow g^b$	$b \xleftarrow{R} \mathbb{Z}_p^*$; $B \leftarrow g^b$
$c \xleftarrow{R} \mathbb{Z}_p^*$; $C \leftarrow g^b$	$c \xleftarrow{R} \mathbb{Z}_p^*$; $C \leftarrow g^b$	$c \xleftarrow{R} \mathbb{Z}_p^*$; $C \leftarrow g^b$
Return $(\overrightarrow{\mathbb{G}}, g, A, B, C)$	$z \leftarrow abc \mod p$; $Z \leftarrow \hat{e}(g,g)^z$	$z \xleftarrow{R} \mathbb{Z}_p^*$; $Z \leftarrow \hat{e}(g,g)^z$
	Return $(\overrightarrow{\mathbb{G}}, g, A, B, C, Z)$	Return $(\overrightarrow{\mathbb{G}}, g, A, B, C, Z)$
proc Finalize(Z)	**proc Finalize**(β')	**proc Finalize**(β')
Return $(Z = \hat{e}(g,g)^{abc})$	Return $(\beta' = 1)$	Return $(\beta' = 1)$

Figure 1.2: Games $\mathbf{Exp}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$, $\mathbf{Exp}^{\text{bddh-0}}_{\mathcal{G},k}(\mathcal{A})$, and $\mathbf{Exp}^{\text{bddh-1}}_{\mathcal{G},k}(\mathcal{A})$ defining the advantage of an adversary $\mathcal{A}$ against the BDH and BDDH problems relative to a pairing parameter generator $\mathcal{G}$ and security parameter k.

for $\beta \in \{0,1\}$. In these games, the procedure **Initialize** first chooses a random generator $g \xleftarrow{R} \mathbb{G}^*$, random integers $a, b, c \xleftarrow{R} \mathbb{Z}_p^*$, and a random element $T \xleftarrow{R} \mathbb{G}_T$. If $\beta = 1$, **Initialize** returns the tuple $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c, \hat{e}(g,g)^{abc})$ to $\mathcal{A}$; if $\beta = 0$, it returns the tuple $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}), g, g^a, g^b, g^c, T)$ instead. The advantage $\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},k}(\mathcal{A})$ is then defined as the probability that game $\mathbf{Exp}^{\text{bddh-0}}_{\mathcal{G},k}(\mathcal{A})$ outputs true minus the probability that game $\mathbf{Exp}^{\text{bddh-1}}_{\mathcal{G},k}(\mathcal{A})$ outputs true. Finally, the BDH and BDDH problems are said to be hard relative to $\mathcal{G}$ if $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},k}(\mathcal{A})$ and $\mathbf{Adv}^{\text{bddh}}_{\mathcal{G},k}(\mathcal{A})$ are negligible functions in k for all PTAs $\mathcal{A}$.

1.4 Example: Public-key encryption

In order to illustrate how provable security can be used in practice, we provide in this section a proof of security for the classical ElGamal public-key encryption scheme [ElG85a] as an example. Towards this goal, we start by recalling the formal definition of a public-key encryption scheme and what it means to be secure. Next, we describe the ElGamal public-key encryption scheme and show that it meets the notion of indistinguishability under chosen-plaintext attacks under the decisional Diffie-Hellman assumption described in Section 1.3.

Syntax. A *public-key encryption* scheme (PKE) is defined by a tuple of algorithms $\mathsf{PKE} = (\mathsf{PG}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ and a message space $\mathcal{M}$, providing the following functionality. Via $pars \xleftarrow{R} \mathsf{PG}(1^k)$, one can run the probabilistic parameter generation algorithm PG to setup the common parameter $pars$ for a given security parameter k. Via $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}(pars)$, a user can run the probabilistic algorithm KeyGen to obtain a pair (pk, sk) of public and secret keys with respect to common parameter $pars$. Via $C \xleftarrow{R} \mathsf{Enc}(pk, m)$, one can send an encrypted message $m \in \mathcal{M}$ to the user with public pk. Finally, via $m \leftarrow \mathsf{Dec}(sk, C)$, the user in possession of the secret key sk and a ciphertext C can run the deterministic decryption algorithm to recover the underlying plaintext m. For correctness, it is required that for all honestly generated keys $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}$, for all messages $m \in \mathcal{M}$, $m = \mathsf{Dec}(sk, \mathsf{Enc}(pk, m))$ holds with all but negligible probability.

Security definition. The now-standard definition of security of PKE schemes, suggested by Goldwasser and Micali [GM84], is indistinguishability under *chosen-plaintext attacks* (IND-CPA). In this security model, the adversary receives the public key of the scheme that he is trying to attack and his goal is to find a pair of messages of the same length whose encryptions he is able to distinguish. Since the adversary is allowed to choose the challenge messages after seeing the public key, this security notion implicitly provides security against key recovery attacks.

The precise definition of IND-CPA security considers the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ described in Figure 1.3. $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ contains three procedures, which are executed with an adversary $\mathcal{A}$ as follows. The procedure **Initialize** generates the common parameter $pars \xleftarrow{R} \mathsf{PG}(1^k)$ and a pair of public and secret keys $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}(pars)$ and returns pk to $\mathcal{A}$. During the execution of the game, the adversary is allowed to make a *single* query (m_0^*, m_1^*) to the **LR** procedure, where $m_0^*, m_1^* \in \{0,1\}^*$ are assumed to have the same length. To answer it, the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ generates a challenge ciphertext $C^* \xleftarrow{R} \mathsf{Enc}(pk, m_\beta^*)$ and gives C^* to $\mathcal{A}$. Eventually, the adversary ends the game by querying the **Finalize** procedure with a guess β' for the bit β used to generate the challenge ciphertext. The advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},k}(\mathcal{A})$ of the adversary $\mathcal{A}$ in breaking the IND-CPA security of PKE is then defined as the probability that game $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{PKE},k}(\mathcal{A})$ outputs true minus the probability that game $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{PKE},k}(\mathcal{A})$ outputs true. Finally, we say that PKE is secure if $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},k}(\mathcal{A})$ is a negligible function in k for all PTAs $\mathcal{A}$.

ElGamal encryption. The ElGamal public-key encryption scheme, described in Figure 1.4, was proposed in [ElG85a]. It can be seen as an adaptation of the Diffie-Hellman key exchange described in Figure 1 on page 5 to the public-key setting by fixing the first message sent by one the parties as the public key for that party.

As in the Diffie-Hellman key exchange, the ElGamal encryption scheme works over a finite cyclic group $\mathbb{G}$ of prime order p obtained via a group generator $\mathcal{G}$. To generate a pair (pk, sk) of public and secret keys, the user chooses a generator g for $\mathbb{G}$ and a random element $x \in \mathbb{Z}_p^*$, computes $X \leftarrow g^x$, and sets $pk = (\mathbb{G}, p, g, X)$ and $sk = (\mathbb{G}, p, g, x)$. To encrypt a message $m \in \mathbb{G}$ to a user with public key $pk = (\mathbb{G}, p, g, X)$, the sender simply chooses a random element $r \in \mathbb{Z}_p^*$

Game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$

proc Initialize(k)	**proc LR**(m_0^*, m_1^*)	**proc Finalize**(β')
$pars \xleftarrow{R} \mathsf{PG}(1^k)$	$C^* \xleftarrow{R} \mathsf{Enc}(pk, m^*_\beta)$	Return $(\beta' = 1)$
$(pk, sk) \xleftarrow{R} \mathsf{KeyGen}(pars)$	Return C^*	
Return pk		

Figure 1.3: Game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{PKE},k}(\mathcal{A})$ for $\beta \in \{0,1\}$ defining the IND-CPA security of a public-key encryption scheme $\mathsf{PKE} = (\mathsf{PG}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

and outputs $(C_1, C_2) = (g^r, m \cdot X^r)$ as the ciphertext. To decrypt it, the user in possession of the secret key $sk = (\mathbb{G}, p, g, x)$ corresponding to $pk = (\mathbb{G}, p, g, X)$ computes $C_2 / {C_1}^x$ to recover the underlying message.

$\mathsf{PG}(1^k)$:	$\mathsf{KeyGen}(pars)$:	$\mathsf{Enc}(pk, m)$:	$\mathsf{Dec}(sk, C)$:
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	parse $pars$ as $(\mathbb{G}, p)$	$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1 \leftarrow g^r$	parse C as (C_1, C_2)
$pars \leftarrow (\mathbb{G}, p)$	$g \xleftarrow{R} \mathbb{G}^*$	$K \leftarrow X^r$	parse sk as $(\mathbb{G}, p, g, x)$
Return $pars$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$C_2 \leftarrow m \cdot K$	$m' \leftarrow C_2 / {C_1}^x$
	$sk \leftarrow (\mathbb{G}, p, g, x)$	Return (C_1, C_2)	Return m'
	$pk \leftarrow (\mathbb{G}, p, g, X)$		
	Return (pk, sk)		

Figure 1.4: The ElGamal public-key encryption scheme [ElG85a], where $\mathcal{G}$ is a group generator.

Security of ElGamal encryption. In order to prove that the ElGamal encryption scheme meets the IND-CPA security notion depicted in Figure 1.3 if the DDH problem is hard with respect to group generator $\mathcal{G}$, we will provide a reduction which relates the advantage of the adversary in breaking IND-CPA security game to the advantage of another adversary in breaking the DDH problem with respect to $\mathcal{G}$. More precisely, we want to prove the following theorem.

Theorem 1.4.1 Let EG refer to the ElGamal PKE scheme in Figure 1.4, let $\mathcal{G}$ be the underlying group generator, let k be the security parameter, and let $\mathcal{A}$ be an adversary against IND-CPA security notion as depicted in Figure 1.3, making at most a single query to the **LR** procedure. Then, there exists an adversary $\mathcal{B}$ against the DDH problem with respect to $\mathcal{G}$, whose running time is that of $\mathcal{A}$ and such that

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{EG},k}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\text{ddh}}_{\mathcal{G},k}(\mathcal{B}).$$

The actual proof of Theorem 1.4.1 is quite simple and uses the fact that, in order to distinguish between the encryption of the challenge messages, the adversary needs to somehow distinguish the value $K = X^r = g^{rx}$ used to hide the message in C_2 from a random element in the group, when given the public key $pk = (\mathbb{G}, p, g, X = g^x)$ and the first part of the ciphertext, $C_1 = g^r$. However, to make this intuition more precise and to illustrate the usefulness of games in security proofs, we will prove Theorem 1.4.1 using a sequence of hybrid games.

Our proof contains a total of 5 games, which are described in Figure 1.5, in which games G_0 and G_4 correspond respectively to the games $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{EG},k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{EG},k}(\mathcal{A})$ of the IND-CPA security definition. Hence, in order to show that the advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{EG},k}(\mathcal{A})$ of $\mathcal{A}$

Game G_0	**Game** G_1	**Game** G_2	**Game** G_3	**Game** G_4
proc Initialize(k)	**proc Initialize**(k)	**proc Initialize**(k)	**proc Initialize**(k)	**proc Initialize**(k)
$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$	$(\mathbb{G}, p) \xleftarrow{R} \mathcal{G}(1^k)$
$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$	$g \xleftarrow{R} \mathbb{G}^*$
$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$	$x \xleftarrow{R} \mathbb{Z}_p^*$; $X \leftarrow g^x$
$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$	$sk \leftarrow (\mathbb{G}, p, g, x)$
$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$	$pk \leftarrow (\mathbb{G}, p, g, X)$
Return pk	Return pk	Return pk	Return pk	Return pk
proc LR(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)	**proc LR**(m_0^*, m_1^*)
$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1^* \leftarrow g^r$	$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1^* \leftarrow g^r$	$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1^* \leftarrow g^r$	$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1^* \leftarrow g^r$	$r \xleftarrow{R} \mathbb{Z}_p^*$; $C_1^* \leftarrow g^r$
$K \leftarrow X^r$	$\boxed{K \xleftarrow{R} \mathbb{G}^*}$	$K \xleftarrow{R} \mathbb{G}^*$	$K \xleftarrow{R} \mathbb{G}^*$	$\boxed{K \leftarrow X^r}$
$C_2^* \leftarrow m_0^* \cdot K$	$C_2^* \leftarrow m_0^* \cdot K$	$\boxed{C_2^* \xleftarrow{R} \mathbb{G}}$	$\boxed{C_2^* \leftarrow m_1^* \cdot K}$	$C_2^* \leftarrow m_1^* \cdot K$
Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)	Return (C_1^*, C_2^*)
proc Finalize(β')	**proc Finalize**(β')	**proc Finalize**(β')	**proc Finalize**(β')	**proc Finalize**(β')
Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$	Return $(\beta' = 1)$

Figure 1.5: Sequence of games for the security proof of the ElGamal PKE scheme described in Figure 1.4, where the rectangular boxes indicate differences with respect to the previous game. $\mathcal{G}$ is the underlying group generator used in the ElGamal PKE scheme.

against EG is negligible for all PTAs $\mathcal{A}$, it suffices to show that the probability that $\mathrm{G}_i^{\mathcal{A}}$ outputs true for $i = 0, \ldots, 4$ does not change significantly.

Proof: Consider the sequence of games depicted in Figure 1.5. By substituting the description of the ElGamal PKE scheme (EG) in Game $\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-}\beta}(\mathcal{A})$ for $\beta \in \{0,1\}$ in Figure 1.3, we have that

$$\begin{aligned} \mathbf{Adv}_{\mathsf{EG},k}^{\text{ind-cpa}}(\mathcal{A}) &= \Pr\left[\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-0}}(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-1}}(\mathcal{A}) = \mathsf{true}\right] \\ &= \Pr\left[\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathrm{G}_4(\mathcal{A}) = \mathsf{true}\right]. \end{aligned} \tag{1.1}$$

We now claim that there exists an adversary $\mathcal{B}_1$ against the DDH problem relative to $\mathcal{G}$ such that $\Pr\left[\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\right] \leq \mathbf{Adv}_{\mathcal{G},k}^{\text{ddh}}(\mathcal{B})$. In order to prove this claim, we build $\mathcal{B}_1$ as follows. Let $((\mathbb{G}, p), g, X, Y, Z)$ be the input that $\mathcal{B}$ receives from the **Initialize** procedure in $\mathbf{Exp}_{\mathcal{G},k}^{\text{ddh-}\beta}(\mathcal{B}_1)$. $\mathcal{B}_1$ then sets $(\mathbb{G}, g)$ as the underlying group for EG and returns $pk = (\mathbb{G}, p, g, X)$ to $\mathcal{A}$ as the output of its **Initialize** procedure of $\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-0}}(\mathcal{A})$. When $\mathcal{A}$ queries its **LR** procedure with a pair of messages (m_0^*, m_1^*), $\mathcal{B}_1$ simulates its behavior by setting $(C_1^*, C_2^*) = (Y, m_0^* \cdot Z)$ and returning it to $\mathcal{A}$. Finally, when $\mathcal{A}$ queries its **Finalize** procedure with a guess bit β', $\mathcal{B}_1$ queries its own **Finalize** procedure with β'.

Given the above description of $\mathcal{B}_1$, it is not hard to see that, when we execute $\mathcal{B}_1$ in $\mathbf{Exp}_{\mathcal{G},k}^{\text{ddh-0}}(\mathcal{B}_1)$, $\mathcal{B}_1$ simulates $\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-0}}(\mathcal{A})$ to $\mathcal{A}$. Likewise, when $\mathcal{B}_1$ interacts with the challenger in $\mathbf{Exp}_{\mathcal{G},k}^{\text{ddh-1}}(\mathcal{B}_1)$, it simulates $\mathbf{Exp}_{\mathsf{EG},k}^{\text{ind-cpa-1}}(\mathcal{A})$ to $\mathcal{A}$. Thus,

$$\begin{aligned} \Pr\left[\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\right] &= \Pr\left[\mathbf{Exp}_{\mathcal{G},k}^{\text{ddh-0}}(\mathcal{B}_1) = \mathsf{true}\right] \\ \Pr\left[\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\right] &= \Pr\left[\mathbf{Exp}_{\mathcal{G},k}^{\text{ddh-1}}(\mathcal{B}_1) = \mathsf{true}\right] \end{aligned}$$

which implies that

$$\Pr\left[\mathrm{G}_0(\mathcal{A}) = \mathsf{true}\right] - \Pr\left[\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\right] \leq \mathbf{Adv}_{\mathcal{G},k}^{\text{ddh}}(\mathcal{B}_1). \tag{1.2}$$

Due to the similarities between games $\mathrm{G}_0(\mathcal{A})$ and $\mathrm{G}_4(\mathcal{A})$ and games $\mathrm{G}_1(\mathcal{A})$ and $\mathrm{G}_3(\mathcal{A})$, we can easily build an adversary $\mathcal{B}_2$ against the DDH problem relative to $\mathcal{G}$ such that

$$\Pr\left[\,\mathrm{G}_3(\mathcal{A}) = \mathsf{true}\,\right] - \Pr\left[\,\mathrm{G}_4(\mathcal{A}) = \mathsf{true}\,\right] \;\leq\; \mathbf{Adv}^{\mathrm{ddh}}_{\mathcal{G},k}(\mathcal{B}_2)\;. \tag{1.3}$$

Finally, we note that the differences between games $\mathrm{G}_1(\mathcal{A})$ and $\mathrm{G}_2(\mathcal{A})$ and between games $\mathrm{G}_2(\mathcal{A})$ and $\mathrm{G}_3(\mathcal{A})$ are purely syntactic since, in all of these cases, C_2^* is a random element in the group which does not depend on the guess bit β used to select the challenge message. Hence,

$$\Pr\left[\,\mathrm{G}_1(\mathcal{A}) = \mathsf{true}\,\right] = \Pr\left[\,\mathrm{G}_2(\mathcal{A}) = \mathsf{true}\,\right] = \Pr\left[\,\mathrm{G}_3(\mathcal{A}) = \mathsf{true}\,\right]. \tag{1.4}$$

The proof of Theorem 1.4.1 follows by combining equations 1.1, 1.2, 1.3, and 1.4 and by noticing that the adversary $\mathcal{B}$ of the theorem statement runs $\mathcal{B}_1$ with probability 1/2 and $\mathcal{B}_2$ with probability 1/2. ▮

Chapter 2

Password-based cryptography

2.1 Introduction

Password-based encrypted key exchange (PAKE) protocols are a particular case of authenticated key exchange protocols in which the secret key or password used for authentication is not uniformly distributed over a large space, but rather chosen from a small set of possible values (a four-digit pin, for example). Since PAKE protocols rely on short and easily memorizable secrets, they also seem more convenient to use as they do not require an additional cryptographic devices capable of storing high-entropy secret keys.

Due to their practicality, password-based key exchange protocols have been very popular over the years. Unfortunately, the vast majority of protocols found in practice do not account for the fact that passwords have low entropy and are often subject to the so-called *dictionary* attacks. These are attacks in which an adversary tries to break the security of a scheme by a brute-force method, by trying all possible combinations of secret keys in a given small set of values (i.e., the dictionary). Although not very effective in the case of high-entropy keys, dictionary attacks can be very damaging when the secret key is a password since the attacker has a non-negligible chance of winning. Such attacks are usually divided in two categories: *off-line* and *online* dictionary attacks.

To address the problem of dictionary attacks, several protocols have been designed to be secure even when the secret key is a password. The goal of these protocols is to restrict the adversary's success to online dictionary attacks only, in which the adversary must be present and interact with the system in order to be able to verify whether its guess is correct. The security in these systems usually relies on a policy of invalidating or blocking the use of a password if a certain number of failed attempts has occurred.

Overview of this chapter. Since the security analysis of PAKE schemes is one of the most challenging aspects in their design, we recall in Section 2.2 the different security notions used in the analysis of PAKE schemes. Next, we continue our review of PAKE schemes by recalling the first seminal work in this area, namely the encrypted key exchange protocol by Bellovin and Merritt [BM92], together with its main variants. As security of existing EKE-based protocols relies on idealized models, such as the random-oracle model, we review in Section 2.4 the main PAKE schemes with a proof of security in the standard model. Then, in Section 2.5, we discuss schemes which remain secure under arbitrary composition and when participants run the protocol with different but possibly related passwords. Finally, in Sections 2.6, 2.7, and 2.8, we consider extensions to the group setting.

2.2 Security models for PAKE

Due to the low entropy of passwords, password-authenticated protocols are always subject to online guessing attacks in which an attacker tries to impersonate one of the parties by simply guessing the correct password and running the protocol. Since these attacks are unavoidable and can succeed with non-negligible probability, standard notions of security for authenticated key exchange protocols cannot be used in this scenario. To tackle this issue and capture the non-negligible success probability of online dictionary attacks, three different types of security models have been proposed in the literature based on the indistinguishability-based model of Bellare and Rogaway [BR94a], on the simulation-based model of Bellare, Canetti, and Krawczyk [BCK98], and on the universal composability of Canetti [Can01]. We now briefly review each of these security models.

2.2.1 Indistinguishability-based model

The first indistinguishability-based security model for PAKE protocols was proposed by Bellare, Pointcheval, and Rogaway (BPR) in [BPR00]. As in [BR94a], each instance of a user is modeled as a stateful oracle with which the adversary can interact via specific oracle queries, which model the adversary's capabilities in a real attack. These queries will usually have the form (U_1, i, U_2, m), meaning that the adversary is sending a message m to instance i of user U_1 claiming that it is from user U_2 and will be answered according to the protocol specification. Since several sessions may be executed concurrently in a real execution of the protocol, several user instances may be active at any given time.

Let U^i denote the i-th instance of a user U and let b be a bit chosen uniformly at random. The standard queries used to model the adversary's capabilities in a real attack are *Execute*, *Send*, *Reveal*, *Corrupt*, and *Test*.

- *Execute*(U_1^i, U_2^j): This query models passive attacks in which the attacker eavesdrops on honest executions between a user instances U_1^i and U_2^j. Its output consists of the messages that would be exchanged between these two instances during an honest execution of the protocol.

- *Send*(U^i, m): This query models an active attack, in which the adversary may intercept a message and then either modify it, create a new one, or simply forward it to the intended participant. Its output is the message that the participant instance U^i would generate upon receipt of message m.

- *Reveal*(U^i): This query models the misuse of session keys by clients. It returns to the adversary the session key of user instance U^i, if the latter is defined.

- *Corrupt*(U): The exact output of this query depends on the corruption model which is being assumed. Usually, this query simply returns the long-lived key of user U, which in the password-based setting means his password. However, in a stronger corruption model, the adversary may also learn the internal states of all instances of user U.

- *Test*(U^i): This query captures the adversary's ability to tell apart a real session key from a random one and returns the session key for instance U^i if $b = 1$ or a random key of the same size if $b = 0$.

While the first four queries serve to model the adversary's capability in a real attack, the last one intends to capture a specific security goal for key exchange protocols, which is the difficulty of distinguishing real session keys from a random one. Nonetheless, in order to define the latter

security goal more precisely, we first need to introduce the notions of *partnering* and *freshness*. While *partnering* defines the set of user instances which should be treated together, *freshness* captures situations in which the adversary can trivially know the value of the session key being held by a user instance.

Partnering. Following [KY03], the notion of partnering is defined via session and partner identifiers. More precisely, let the session identifier sid_U^i be a function of all the messages sent and received by user instance U^i as specified by the key exchange protocol. Let the partner identifier pid_U^i be the set of users with whom user instance U^i wishes to establish a common secret key, including U^i himself. We say that U_1^i and U_2^j are partners if and only if they have the same session and partner identifiers.

Freshness. A user instance is said to be fresh if neither this instance nor its partner have been corrupted and if no *Reveal* query has been asked to it or to its partner.

Now that we have defined these properties, we can state the two main properties that a secure PAKE protocol should satisfy: *correctness* and *semantic security*.

- **Correctness.** For a protocol to be correct, it should always be the case that, whenever two instances U_1^i and U_2^j are partnered and have successfully completed the protocol, both instances should hold the same non-null session key.

- **Security.** Consider an execution of the PAKE protocol P by a polynomial-time adversary, in which the latter is given access to the *Reveal*, *Execute*, *Send*, *Corrupt* and *Test* oracles and asks a single *Test* query to a *fresh* instance, and outputs a guess bit b'. Let b be the hidden bit used by the *Test* oracle and assume that user passwords are chosen uniformly from a dictionary space Dict. We say that the PAKE protocol P is secure if the probability that $b' = b$ is only negligibly larger than $O(q/|\mathsf{Dict}|) + 1/2$, where q is number of different protocol instances to which the adversary has asked *Send* queries.

Remark 2.2.1 We note that the factor $1/2$ in the security definition is to disregard as unsuccessful attacks in which the adversary simply guesses the value of the hidden bit b uniformly at random. Likewise, the factor $O(q/|\mathsf{Dict}|)$ is to account for the fact that the adversary can succeed with probability $q/|\mathsf{Dict}| + 1/2$ by simply guessing the password of a given user and playing its role via *Send* queries.

Contributions. In [AFP05, AFP06], Fouque, Pointcheval, and I extended the security notion above to the scenario in which the adversary is allowed to ask multiple *Test* queries. As we show in these articles, one of the advantages of the new notion is that it provides security proofs with better concrete bounds when the PAKE protocol is used inside other protocols. This is due to the fact all session keys can be replaced with random ones in a single hybrid step within these security reductions. Though not important for the standard AKEs, such factors are extremely important in the password-based scenario due to low entropy of passwords.

2.2.2 Simulation-based model

The first simulation-based security model for PAKE protocols was proposed by Boyko, MacKenzie, and Patel (BMP) in [BMP00], based on the formal security models by Shoup [Sho99] and by Bellare, Canetti, and Krawczyk [BCK98]. As in [Sho99], security is defined using an ideal system, which models the service being performed (PAKE in this case), and a real system, which describes the actual environment in which the protocol is executed. Intuitively, in order to prove the security of a PAKE scheme, it suffices to show that anything that an adversary can do in a real system can also be done in the ideal system.

As in the indistinguishability-based model, each user can have several instances associated to him and the adversary interacts with the system via oracle queries. Additionally, each user assumes a particular *role* in each session, depending on whether it is going to open itself for connection or whether it is going to connect to another instance. Unlike the BPR model, there is a new trusted entity called the *ring master*, which is responsible for keeping track of the session keys that are established among user instances, and an environment variable R modeling he information shared by users in higher-levels protocols.

Ideal world. To model the PAKE service in the ideal world, the following types of queries are available to the adversary.

- *InitUser*(U, id): This query allows the adversary to assign the string id to user U and models the initialization of the latter. Upon receiving this query, for each user U_i with identity id_i in the system, the ring master chooses a password pw_i at random from the dictionary and assigns it to the pair (id, id_i).
- *SetPassword*(U_i, id', pw): This query allows the adversary to set the password for the pair (id, id') to pw as long as id' has not been assigned to any user.
- *InitInstance*$(U_i^j, role, \mathsf{pid})$: This query assigns the role for the j-th instance of user U_i in a session with user pid, if the latter is defined.
- *TerminateInstance*(U_i^j): This query forces instance U_i^j to terminate its session.
- *TestPassword*(U_i^j, pw): This query models a password guess by the adversary with respect to user instance U_i^j. This query returns either `true` or `false` to the adversary depending on whether the guess is correct. The adversary is only allowed to query this oracle once per user instance.
- *StartSession*(U_i^j, opt): This query forces the construction of a session key K_i^j for U_i^j. If U_i^j is open for connection, untested (no query *TestPassword* has been asked of it), and both U_i^j and its partner have not been corrupted (not initialized via *InitUser*), then the ring master chooses K_i^j uniformly at random from the proper distribution. Likewise, if U_i^j is connecting to another instance, untested, and both U_i^j and its partner have not been corrupted, then K_i^j is set to key value assigned to its partner. If U_i^j is corrupted or if a successful *TestPassword* has been asked of it, then the ring master sets K_i^j to value specified in opt.
- *Application*(U_i^j, f): This query models leakage of the session key information through the use of the latter in a real protocol. It enables the adversary to learn an efficiently computable function f about the environment variable and the session keys.

Real world. In the real world, user instances are defined as state machines with access to the user's id, pid, password, and private random inputs. Each user instance has an initial state and only updates its state once it receives an input message, at which point it generates a reply message. When generating the response, a user instance also updates its status to indicate that it is prepared to receive another message (*continue*), that it has finished and generated a session key (*accept*), or that it has finished unsuccessfully (*reject*).

In addition to the queries *InitUser*, *InitInstance*, *SetPassword*, and *Application* used in the ideal world, the adversary in the real world also has access to a message delivery query *MsgDelivery*. Moreover, if the proof of security requires an idealized model, such as the random-oracle model [BR93], the adversary is also given access to the latter.

Secure PAKE. Now that we have defined the ideal and real worlds, we can state the two main properties that a secure PAKE protocol should satisfy in the BMP model: *completeness* and *security*.

- **Completeness.** For any real-world adversary that faithfully follows the protocol, delivering messages between two user instances with complementary roles and identities, both user instances should accept and hold the same non-null session key.
- **Security.** For any efficient real-world adversary, there exists an efficient ideal-world adversary such that the transcripts of the adversary's operations in the ideal and real worlds are computationally indistinguishable.

2.2.3 Universally-composable-based model

Even though indistinguishability-based and simulation-based security models provide a security level that is sufficient for most applications, they fail to consider arbitrary compositions with other protocols or some realistic scenarios such as participants running the protocol with different but possibly related passwords. To overcome these deficiencies, Canetti, Halevi, Katz, Lindell, and MacKenzie [CHK+05] proposed an ideal functionality for PAKE protocols in the universal composability (UC) framework [Can01, CK02], which makes no assumption on the distribution on passwords used by the protocol participants. In particular, the model does not assume independence between the passwords of different parties.

The $\mathcal{F}_{pwKE}$ ideal functionality. In the UC model, the aim is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in arbitrary environments. This model relies on the indistinguishability between two worlds, the ideal world and the real world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, that can be thought of as a trusted party in the context of multi-party computation. In order to compute a function f, this functionality interacts with n players, but there is no communication among the players themselves. These players give their inputs to $\mathcal{F}$, which in turn give them back their outputs. $\mathcal{F}$ ensures that the computation is correct and that the players learn nothing more than their own inputs and outputs. The security is then guaranteed since an adversary $\mathcal{A}$ can only learn and thus modify data of corrupted players.

In order to prove that a protocol P realizes $\mathcal{F}$, one considers an environment $\mathcal{Z}$ that provides inputs to the players and plays the role of a distinguisher between the real world (with players and a real adversary that can control some of them and also the communication among them) and the ideal world (with dummy players interacting only with the ideal functionality $\mathcal{F}$, and a simulated adversary also interacting with $\mathcal{F}$). The protocol P is then said to realize $\mathcal{F}$ if for all polynomial adversary $\mathcal{A}$, there exists a polynomial simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the two worlds with a significant advantage.

In Figure 2.1, we recall the PAKE ideal functionality first described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted and if they share the same password, then they both end up with the same uniformly-distributed session key, and the adversary learns nothing about it (except that it was indeed generated). On the other hand, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), then it is granted the right to fully determine its session key. Note that as soon as a party is corrupted, the adversary learns its key: There is in fact nothing lost by allowing it to determine the key.

In addition, the players become aware of a failed attempt of the adversary at guessing a password. This is modeled by marking the session as `interrupted`. In this case, the two players are given independently-chosen random keys.

The functionality $\mathcal{F}_{pwKE}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving a query (`NewSession`, $sid, P_i, P_j, pw, role$) from party P_i:**

  Send (`NewSession`, $sid, P_i, P_j, role$) to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record `fresh`.

- **Upon receiving a query (`TestPwd`, sid, P_i, pw') from the adversary S:**

  If there is a record of the form (P_i, P_j, pw) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If $pw \neq pw'$, mark the record `interrupted` and reply with "wrong guess".

- **Upon receiving a query (`NewKey`, sid, P_i, sk) from the adversary S:**

  If there is a record of the form (P_i, P_j, pw), and this is the first `NewKey` query for P_i, then:

  - If this record is `compromised`, or either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output (sid, sk') to P_i.
  - In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

  Either way, mark the record (P_i, P_j, pw) as `completed`.

Figure 2.1: The password-based key-exchange functionality $\mathcal{F}_{pwKE}$

A session that is nor `compromised` nor `interrupted` is called `fresh`. In such a case, the two parties receive the same, uniformly distributed session key if they execute the protocol with the same password.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password is used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

Contributions. In [ACCP08], Catalano, Chevalier, Pointcheval, and I extend the above definition to the group setting and define an ideal functionality for password-based group key exchange with explicit authentication and contributiveness in the UC framework. As with previous definitions in the same framework, our definitions do not assume any particular distribution on passwords or independence between passwords of different parties. Additionally, the new contributiveness security notion captures security against insider adversaries and guarantees the adversary cannot bias the distribution of the session key as long as a few players involved in the protocol are honest.

2.3 The encrypted key exchange protocol and its variants

The seminal work in the area of password-based key exchange is the encrypted key exchange (EKE) protocol of Bellovin and Merritt [BM92] (see Figure 2.2). In their protocol, two users execute an encrypted version of the Diffie-Hellman key exchange protocol, in which each flow is encrypted using the password shared between these two users as the symmetric key. Intuitively, since the elements to which the encryption function is applied are chosen uniformly at random from the underlying group, an adversary eavesdropping on the communication cannot learn any additional information which would allow him to perform an off-line dictionary attack.

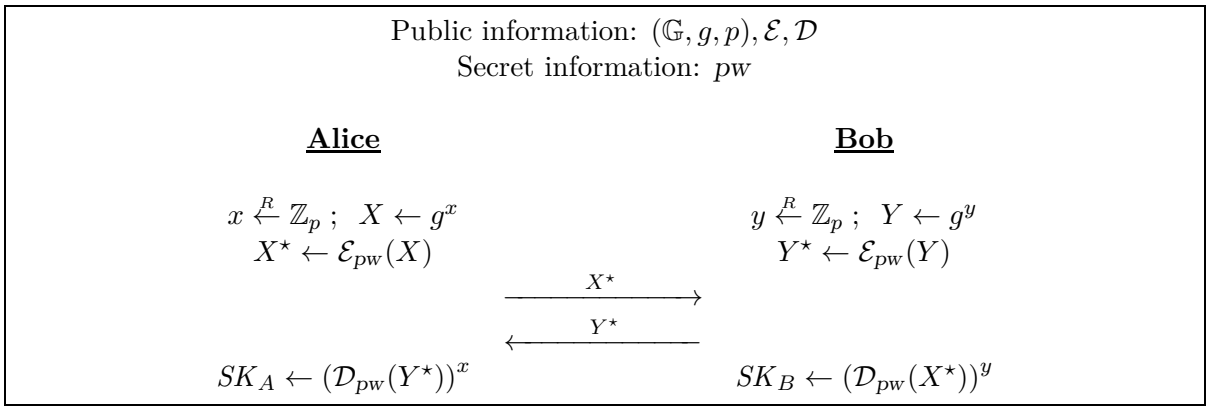


Figure 2.2: The encrypted key exchange protocol [BM92]. The protocol uses symmetric encryption and decryption algorithms $\mathcal{E}$ and $\mathcal{D}$ and works over a finite cyclic group $\mathbb{G}$ of prime order p generated by an element g.

Due to the simplicity of the EKE protocol, several other protocols were soon proposed in the literature based on it [BM93, Luc97, Jab97, STW95]. Unfortunately, due to the lack of a proper security model for the analysis of PAKE schemes, these protocols were only heuristically secure.

It was only in 2000 that Bellare, Pointcheval, and Rogaway [BPR00], as well as Boyko, MacKenzie, and Patel [BMP00], proposed security models for PAKE schemes and proved variants of the EKE protocol, under ideal assumptions, such as the random-oracle model [BR93]. In addition to these, several other protocols were proposed in the literature based on EKE protocol [BCP03, BCP04, Mac02, AP05], each with its own instantiation of the encryption function. Currently, the simple password-authenticated key exchange protocol in Appendix A (to which we refer as SPAKE) is among the most efficient PAKE schemes based on the EKE protocol.

The SPAKE scheme is a variation of the EKE protocol in Figure 2.2, in which the encryption function $\mathcal{E}_{pw}(.)$ is replaced with a simple one-time pad function. More specifically, whenever a user **Alice** wants to send the encryption of a value $X \in \mathbb{G}$ to a user **Bob**, it does so by computing $X \cdot {h_1}^{pw}$, where h_1 is an element in $\mathbb{G}$ associated with user **Alice** and the password pw is assumed to be in $\mathbb{Z}_p$. The session identifier is defined as the transcript of the conversation between **Alice** and **Bob** together with their identities, and the session key is set to be the hash (random oracle) of the session identifier, the password pw, and the Diffie-Hellman key. The full description of SPAKE is given in Figure 2.3.

As we show in Appendix A, SPAKE is a secure PAKE scheme in the random-oracle model according to the definition in Section 2.2.1 if the computational Diffie-Hellman problem is intractable in $\mathbb{G}$.

Contributions. In addition to proposing SPAKE in [AP05] (see Appendix A), Pointcheval and I also proposed in the same paper a variant of SPAKE in which the password pw is *not* an input

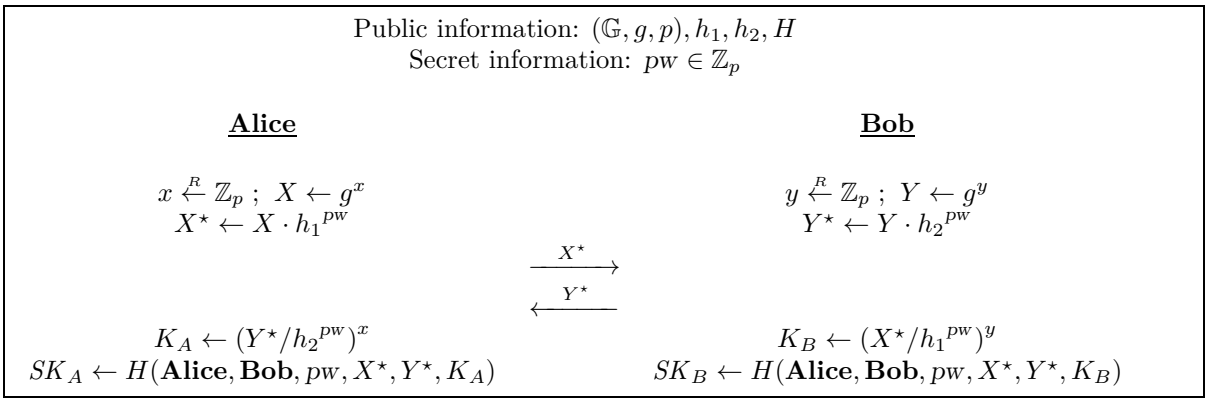


Figure 2.3: SPAKE: A simple password-based key exchange protocol [AP05]. SPAKE works over a finite cyclic group $\mathbb{G}$ of prime order p generated by an element g.

to the hash function H used to derive the session key. While the latter variant is more suitable to scenarios in which the password is shared across several servers, its security only holds in the non-concurrent scenario. The proof of security for both protocols is in the random-oracle model and based on hardness of the computational Diffie-Hellman problem. Moreover, they only require a single random-oracle instance.

In [ACP05], Chevassut, Pointcheval, and I considered a related PAKE protocol, called AuthA, and proved its security even when adversaries are allowed to corrupt users and learn their passwords, which was a known open problem at the time.

In [ABC+06], together with Bresson, Chevassut, Möller, and Pointcheval, we showed how to design an efficient and provably secure PAKE scheme specifically for the TLS (Transport Layer Security) protocol. The goal was to provide a technique that allows users to employ (short) passwords to securely identify themselves to servers. In addition to its provable security, our protocol is also believed to be free from patent and licensing restrictions based on an analysis of relevant patents in the area.

2.4 PAKE protocols in the standard model

Even though EKE-based protocols are extremely efficient and easy to use, their security relies fundamentally on a heuristic assumption, namely the random-oracle model, in which hash functions are assumed to behave as a random oracle. Unfortunately, the random-oracle model is known not to be sound [CGH98]. More precisely, there are several examples of schemes [CGH98, Nie02, GK03, BBP04] that can be proven secure in the random-oracle model and for which there does not exist any concrete instantiation of the hash function for which the scheme remains secure. Hence, it is an important security goal to design schemes which do not rely on any idealized model such a random-oracle model.

The first protocols whose security proof did not rely on any idealized model were proposed by Katz, Ostrovsky, and Yung (KOY) [KOY01] based on the decisional Diffie-Hellman assumption and by Goldreich and Lindell [GL01], who proposed a solution based on general assumptions. While the former KOY protocol assumed the existence of a common reference string, the protocol by Goldreich and Lindell did not rely on any trusted setup assumption. Later, Gennaro and Lindell [GL03] abstracted and generalized (under various indistinguishability assumptions) the KOY protocol using the concept of smooth projective hash functions [CS02], which became the basis of several other protocols [BCL+05, AP06, BGS06, ACP09] in the literature. To

understand how the Gennaro-Lindell protocol works, let us first review the concept of smooth projective hash functions.

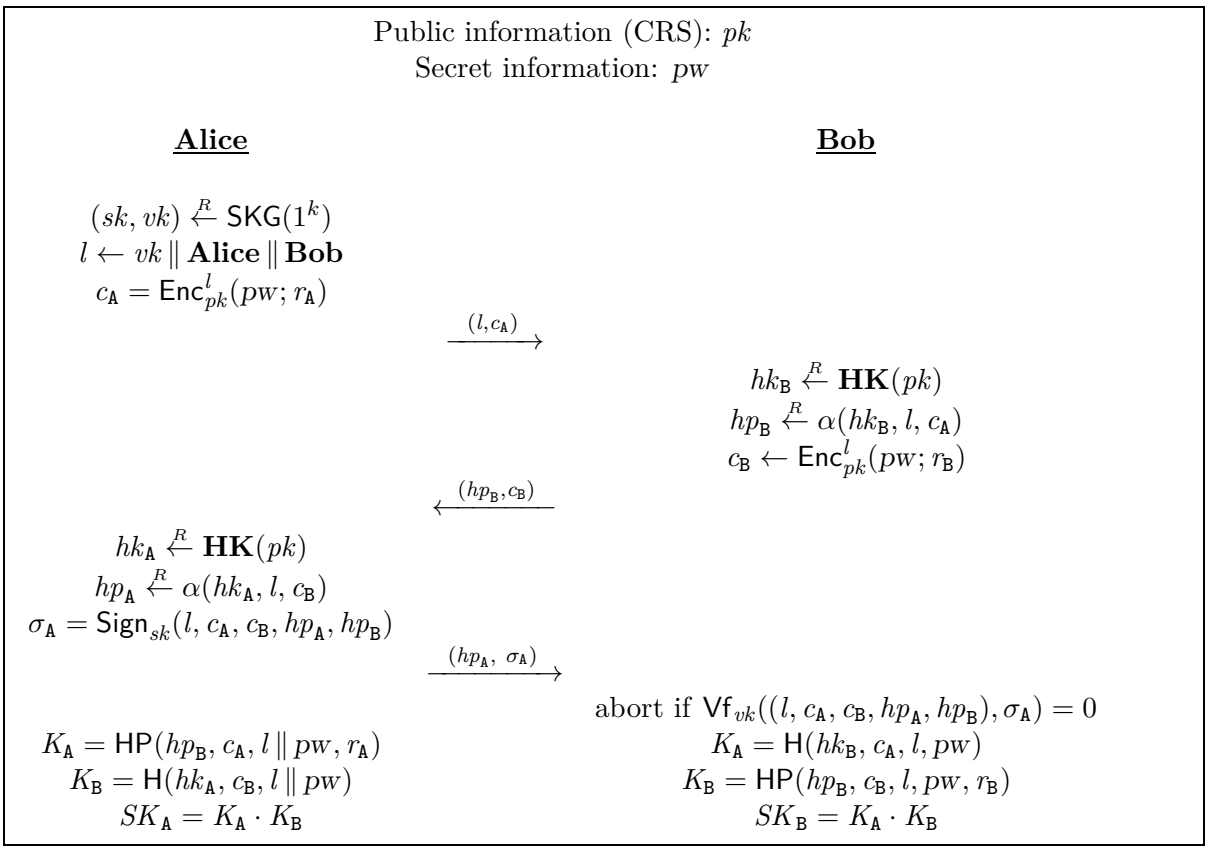


Figure 2.4: An overview of the Gennaro-Lindell PAKE protocol [GL03]. $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption algorithms of a labeled public-key encryption scheme [Sho04]. $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Vf})$ are the key generation, signing, and verification algorithms of a one-time signature scheme [EGM96]. $(\mathsf{HK}, \alpha, \mathsf{H}, \mathsf{HP})$ are the key generation, key projection, hashing, and projected hashing algorithms of a family of smooth projective hash functions for the language L consisting of triples $\{(c, \ell, pw)\}$ such that c is an encryption of the password pw with label ℓ.

Smooth projective hash functions. One of the main tools used in the Gennaro-Lindell (GL) protocol is the notion of smooth projective hash functions (SPHF, [CS02, GL03]), which can be seen as special type of zero-knowledge proof system for an NP language. More precisely, the definition of SPHF requires the existence of a domain X and an underlying NP language L such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. For instance, in the particular case of the PAKE scheme in [CHK$^+$05], the language L is defined as the set of triples $\{(c, \ell, pw)\}$ such that c is an encryption of the password pw with label ℓ under a public key given in the common reference string (CRS). The semantic security of the encryption scheme guarantees computational indistinguishability between elements from L and elements from X.

One of the key properties that make SPHF so useful is that, for a point $x \in L$, the hash value can be computed using either a *secret* hashing key hk, or a *public* projected key hp (depending on x [GL03] or not [CS02]) together with a witness w to the fact that $x \in L$. Another important property of these functions is that, given the projected key hp, their output is uniquely defined

for points $x \in L$ and statistically indistinguishable from random for points $x \in X \setminus L$. Moreover, without the knowledge of the witness w to the fact that $x \in L$, the output of these functions on x is also pseudo-random.

Overview of the GL protocol. Now that we have informally introduced the SPHF concept, we can finally review the GL PAKE protocol, whose detailed description is given in Figure 2.4. At a high level, the players in the GL protocol exchange CCA-secure encryptions of the password, under the public-key found in the common reference string, and then compute the session key by combining smooth projective hashes of the two password/ciphertext pairs. More precisely, the players first exchange ciphertexts consisting of encryption of their respective passwords with respect to the label ℓ containing their identities and the verification key for a one-time signature scheme. Next, each player chooses a hashing key for a smooth projective hash function for the language $\{(\mathsf{Enc}^l_{pk}(pw), \ell, pw)\}$ and sends the corresponding projected key to the other player. Each player can thus compute the output of its own hash function with the help of the hashing key, and the output of the other one using its knowledge of the randomness that was used to generate the ciphertext of the password. To avoid attacks in which the adversary generates new projection keys without modifying the corresponding ciphertexts and projection keys, **Alice** also signs the transcript of the conversation.

To understand informally why this protocol is secure, first consider the case in which the adversary plays a passive role. In this case, the pseudo-randomness property of the smooth hash function ensures that the value of the session key will be computationally indistinguishable from uniform since the adversary does not know the randomness that was used to encrypt the password. Now imagine the case in which the adversary provides the user with an encryption of the wrong password. In this case, the security of the protocol will rely on the smoothness of the hash functions, which ensures that the session key will be random and independent of all former communication. Thus, in order to be successful, the adversary has to generate the encryption of the correct password. To do so, the adversary could try to copy or modify existing ciphertexts. Since the encryption scheme is CCA-secure, and thus non-malleable, modifying is not really a possibility. Copying does not help either since either the label used for encryption will not match (making the session key look random due to the smoothness property) or the signature will be invalid (in the case where the adversary changes the projection keys without changing the label and hence the verification key). As a result, the only successful strategy left for the adversary is essentially to guess the password and perform the trivial online dictionary attack, as desired.

2.5 PAKE protocols in the UC model

Most of the existing PAKE protocols, including the ones mentioned so far, have proofs either in the indistinguishability-based security model of Bellare, Pointcheval, and Rogaway (BPR) or in the simulation-based of Boyko, MacKenzie, and Patel (BMP). As we already mentioned in Section 2.2.3, though these models provide a security level that is sufficient for most applications, they fail to consider some realistic scenarios such as participants running the protocol with different but possibly related passwords. To surmount these deficiencies, Canetti Halevi, Katz, Lindell, and MacKenzie [CHK+05] proposed an ideal functionality for PAKE protocols in the UC framework which makes no assumption on the distribution on passwords used by the protocol participants.

The CHKLM protocol. As noted by Canetti *et al* in [CHK+05], the KOY/GL protocol is not known to achieve UC security: the main issue is that the ideal-model simulator must be able to extract the password used by the adversary. One could think that, since the simulator has control over the common reference string and knows the private keys associated with the public

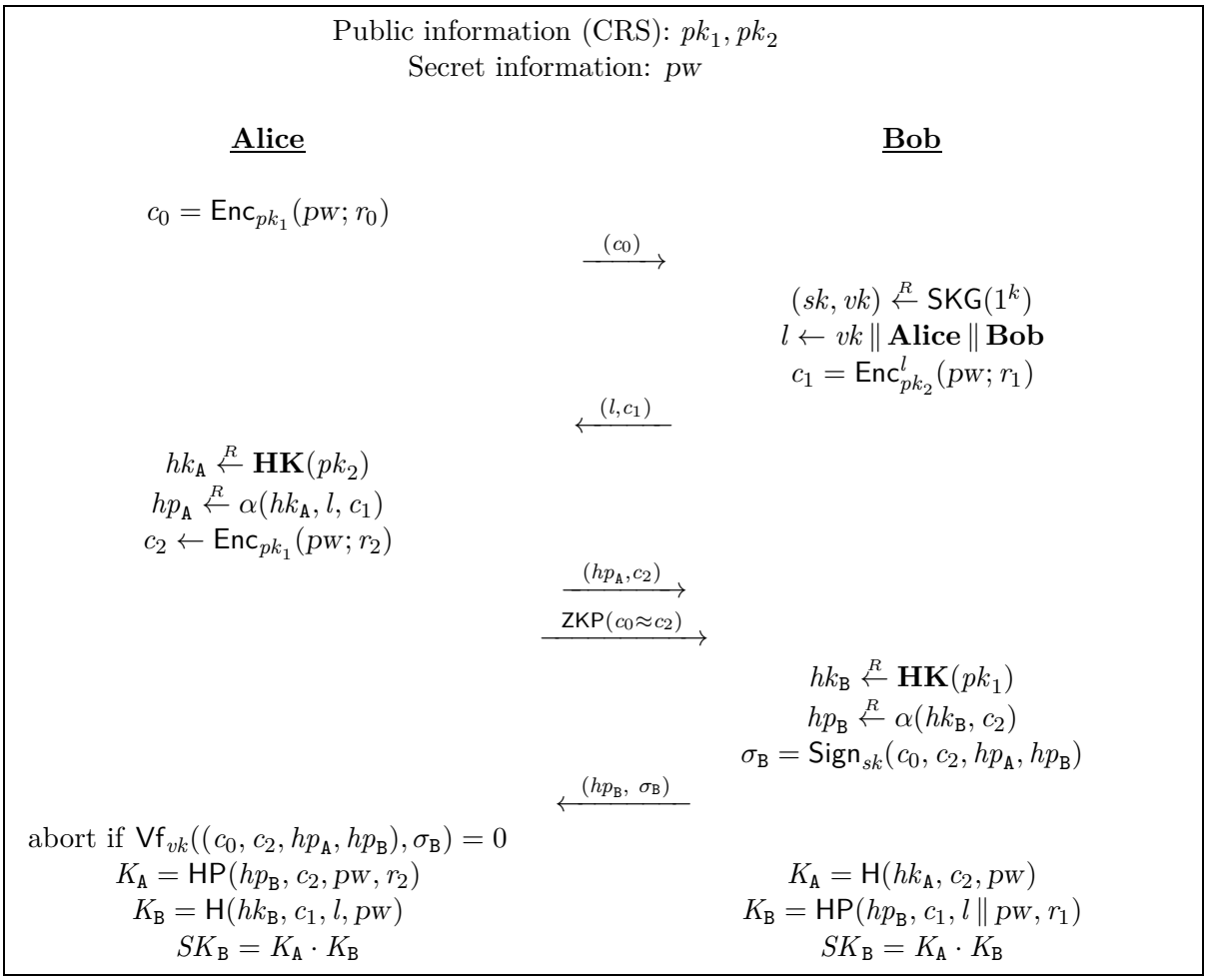


Figure 2.5: An overview of the PAKE protocol by Canetti et al. [CHK+05]. $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption algorithms of a labeled public-key encryption scheme [Sho04]. $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Vf})$ are the key generation, signing, and verification algorithms of a one-time signature scheme [EGM96]. $(\mathsf{HK}, \alpha, \mathsf{H}, \mathsf{HP})$ are the key generation, key projection, hashing, and projected hashing algorithms of a family of smooth projective hash functions for the language L consisting of triples $\{(c, \ell, pw)\}$ such that c is an encryption of the password pw with label ℓ. $\mathsf{ZKP}(c_0 \approx c_2)$ is a simulation-sound zero-knowledge proof [Sah99] that c_0 and c_2 are encryptions of the same value.

keys, it can decrypt all ciphertexts sent by the adversary and recover its password. Yet, this does not seem to be sufficient. In the case where the adversary begins to play (i.e., it impersonates the client), everything works well: The simulator decrypts the ciphertext generated by the adversary and can recover the password used by the latter. If the guess of the adversary is incorrect (that is, the password is the wrong one), then the smoothness of the hash functions leads to random independent session keys. Otherwise, if the guess is correct, the execution can continue as an honest one would do (the simulator has learned which password to use).

Let us now consider the case in which the simulator has to start the game, on behalf of the client. Here, the simulator needs to send an encryption of the password before having seen anything coming from the adversary. As mentioned above, it can recover the password used by the adversary as soon as the latter has sent its value, but this may be too late in this

case. If it turns out that the guess of the adversary is incorrect, then the smoothness property guarantees the statistical independence of the session keys. Conversely, when the adversary's guess is correct, the simulator is stuck with an incorrect ciphertext and will not be able to predict the value of the session key.

To tackle this problem, the authors of [CHK+05] provided a new scheme based on the GL construction [GL03] that securely realizes the ideal functionality for PAKE under static corruptions. In their protocol (to which we refer as CHKLM), which is described in Figure 2.5, the client first sends a pre-flow which also contains an encryption of the password. The server then sends its own encryption, and finally the client sends another encryption (this time the simulator is able to use the correct password, recovered from the value sent by the adversary), as well as a zero-knowledge proof claiming that both ciphertexts are consistent and encrypt the same password. The first flow is never used in the remaining of the protocol. This solves the problem since the simulator is of course able to give a valid proof of a false statement and the first flow is never used afterwards. Moreover, the resulting scheme is quite efficient and enjoys several nice properties such as security under arbitrary compositions with other protocols.

Adaptive security. As mentioned above, the CHKLM protocol is only known to be secure in the presence of static adversaries, when the set of corrupted players is known in advance. However, in reality, the adversary may be able to corrupt parties adaptively and learn their internal states.

To address the issue of adaptive security, Barak, Canetti, Lindell, Pass, and Rabin (BCLPR) proposed in [BCL+05] a simple and intuitive construction that uses general techniques from multi-party computation. Their construction works in two phases. In the *link initialization* phase, each user first generates a fresh pair of signing and verification keys for a strongly unforgeable signature scheme and then broadcasts the verification key to the other users. After receiving the verification keys of all users, each user then signs the sequence of keys received together with the identities of the users and broadcasts it to the other users. Once all signatures are received, each user verifies that all signatures refer to the same set of users. If all the checks succeed, then the users can use their corresponding keys to set up an authenticated channel among themselves. Once the link initialization phase is over, the users proceed to the *secure computation* phase in which they run a generic multi-party computation protocol over the authenticated channels.

As shown in [BCL+05], an adaptively-secure PAKE protocol can be easily obtained in the common reference string (CRS) model via the generic construction above by instantiating it with a multi-party computation protocol for the functionality that returns a long random key to each party if both of their inputs are the same and $\perp$ otherwise. Unfortunately, due to its generality, their protocol is quite inefficient.

In order to understand the difficulties involved in design an efficient adaptively secure PAKE scheme in the standard model, let us consider the case of the CHKLM scheme [CHK+05], described in Figure 2.5. In their protocol, the client needs to to remember the randomness used in the pre-flow message c_0 in order to later prove in zero-knowledge that the password encrypted in c_2 is equal to the one encrypted in c_0. Though sufficient to provide UC security with respect to static adversaries, this modification does not seem to work when dealing with adaptive adversaries. This is because the simulator cannot correctly open the commitment when the adversary corrupts the client after the pre-flow has been sent. A similar remark applies to the case in which the server gets corrupted after sending its first message.

Overview of the ACP protocol. To get around the above problem, Chevalier, Pointcheval and I take a different approach in [ACP09], whose full version can be found in Appendix B. In our protocol, we use the GL PAKE protocol (Figure 2.4) as a starting point and replace the encryption scheme with a non-interactive committing primitive with extraction and equivocation

capabilities. Since the commitment scheme that we use allows for extraction and equivocation at any moment, the simulator can provide a consistent view to the adversary. As this change also impacts the language used by a family of smooth projective hash functions, we also provide a new instantiation of the latter. The resulting scheme, with a few minor technical modifications, is described in Figure 2.6.

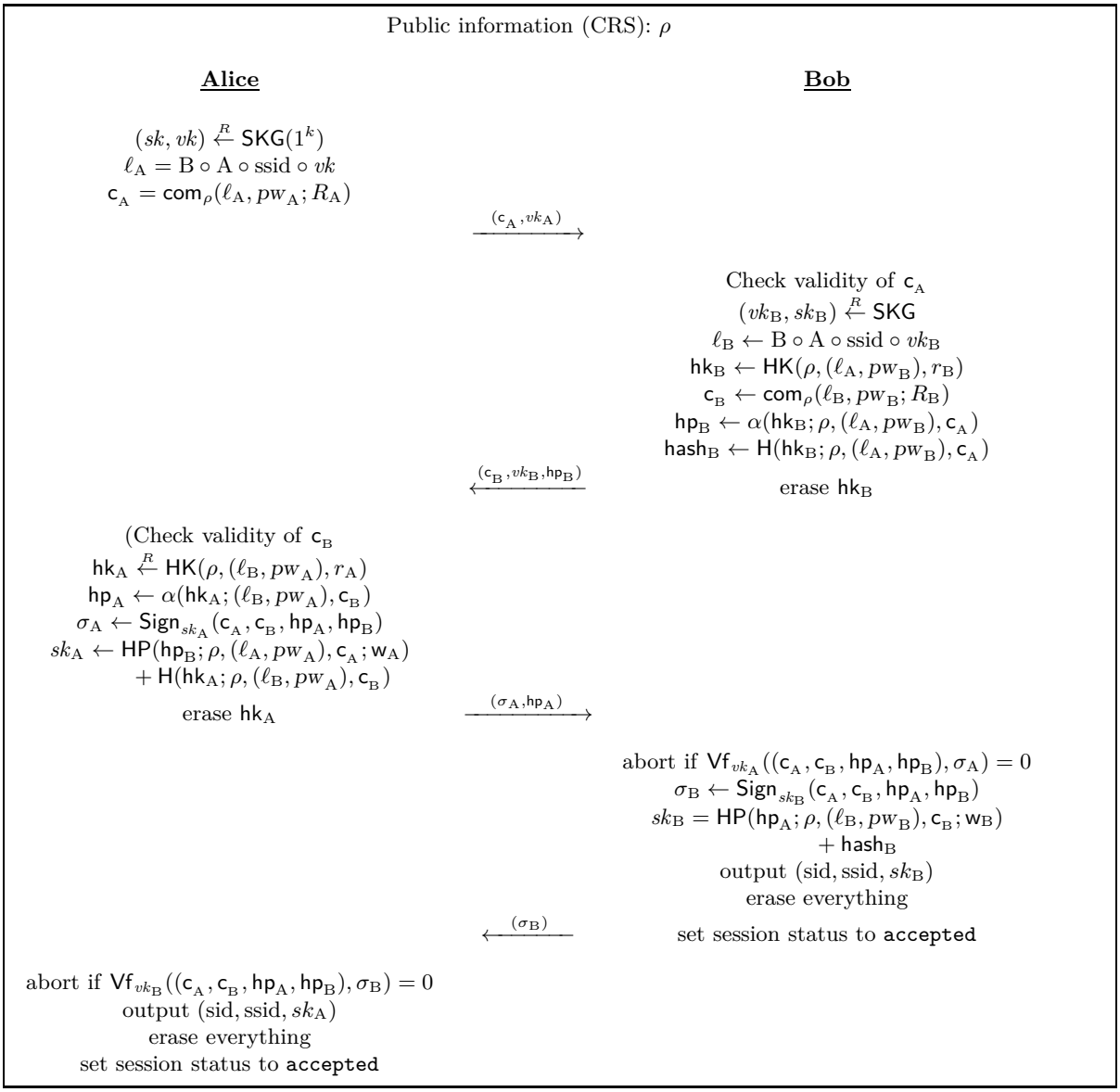


Figure 2.6: The PAKE protocol in Appendix B for players (**Alice**, ssid), with index A and password pw_A and (**Bob**, ssid) with index B and password pw_B. At the end of rounds 1 and 2, the players will erase the part of R_A and R_B which is not needed in the following rounds, keeping only w_A and w_B. $(\mathsf{com}, \mathsf{decom})$ are the commitment and decommitment algorithms of an equivocable and conditionally extractable commitment scheme [ACP09]. $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Vf})$ are the key generation, signing, and verification algorithms of a one-time signature scheme [EGM96]. $(\mathsf{HK}, \alpha, \mathsf{H}, \mathsf{HP})$ are the key generation, key projection, hashing, and projected hashing algorithms of a family of smooth projective hash functions for the language consisting of triples $\{(\mathsf{com}(\ell, pw), \ell, pw)\}$.

Security of the ACP protocol. Let com be the non-malleable (conditionally) extractable

and equivocable committing scheme as described in Appendix B, $\mathcal{H}$ be a family of smooth hash functions with respect to this commitment, and $\mathcal{SIG}$ be a one-time signature scheme. Denote by $\widehat{\mathcal{F}}_{pwKE}$ the multi-session extension of the functionality $\mathcal{F}_{pwKE}$ described in Section 2.2.1, and let $\mathcal{F}_{CRS}$ be the ideal functionality that provides a common reference string ρ to all parties, where ρ are the common parameters for com. Then, as shown in [AP06], the protocol in Figure 2.6 securely realizes $\widehat{\mathcal{F}}_{pwKE}$ in the $\mathcal{F}_{CRS}$-hybrid model, in the presence of adaptive adversaries.

Contributions. In addition to the construction in Appendix B, Catalano, Chevalier, Pointcheval and I provided in [ACCP08] an alternative and more efficient construction of an adaptively-secure PAKE protocol in the UC model. Even though the scheme is almost as efficient as previous EKE-based PAKE constructions, the proof of security relies on the random-oracle and ideal-cipher models.

2.6 PAKE protocols in the group setting

In order to provide support for collaborative and distributed applications, authenticated key exchange has also been considered in the group setting. In these protocols, the goal is to provide a group of users communicating over an insecure channel with an authenticated session key which they can use to secure that subsequent communication. Although protocols for authenticated group Diffie-Hellman key exchange protocols seem to be a natural mechanism for supporting these applications, they rely on the use of public key infrastructures (PKI). To avoid the use of PKIs, Bresson, Chevassut, and Pointcheval [BCP02b, BCP07] considered password-based authentication in the group setting and showed how to adapt their group Diffie-Hellman protocol [BCPQ01] to the password-based scenario. The resulting protocol is reasonably efficient in terms of computation and proven secure in the random-oracle model based on the computational Diffie-Hellman assumption. However, as the original protocol [BCPQ01], the total number of communication rounds is linear in the number of players, making their scheme impractical for large groups.

In order to reduce the total number of communication rounds and improve the scalability of password-based protocols in the group setting, Bresson, Chevassut, Pointcheval and I proposed a new password-authenticated group key exchange (GPAKE) protocol in [ABCP06], which is reproduced in Appendix C. Our new protocol is based on the group key exchange (GKE) protocol by Burmester and Desmedt [BD94, BD05] and provably-secure in the random-oracle and ideal-cipher models, under the Decisional Diffie-Hellman assumption. In addition to being provably secure, the new protocol is also very efficient and fully scalable since it only requires four rounds of communication and four multi-exponentiations per user. To better understand how our constant-round GPAKE scheme works, let us first recall the Burmester-Desmedt GKE protocol [BD94, BD05].

Let $\mathbb{G}$ be a finite cyclic group of prime order p generated by an element g, in which the Decisional Diffie-Hellman (DDH) assumption described in Section 1.3 holds. That is, given elements g, g^a, and g^b in $\mathbb{G}$, no PPT adversary should be able to distinguish g^{ab} from g^c with non-negligible probability, when a, b, c are chosen uniformly at random from $\mathbb{Z}_p$. Let n be the number of players in the group key exchange protocol and assume that all the indices are taken modulo n. The Burmester-Desmedt protocol, whose description is provided in Figure 2.7, works in two rounds. First, each player P_i chooses a random exponent $x_i \in \mathbb{Z}_p$ and broadcasts $X_i = g^{x_i}$. Next, after receiving all first-round messages, each player P_i computes the keys $K_i = X_{i-1}^{x_i}$ and $K_{i+1} = X_{i+1}^{x_i}$ that it shares with its predecessor P_{i-1} and successor P_{i+1}, and broadcasts $Z_i = K_{i+1}/K_i$. Finally, each player P_i computes his session key as $SK_i = \prod_{j=1}^{n} K_j = K_i^n Z_i^{n-1} Z_{i+1}^{n-2} \cdots Z_{i+n-2}$ using the values $Z_1, \ldots, Z_n$ received during the second round.

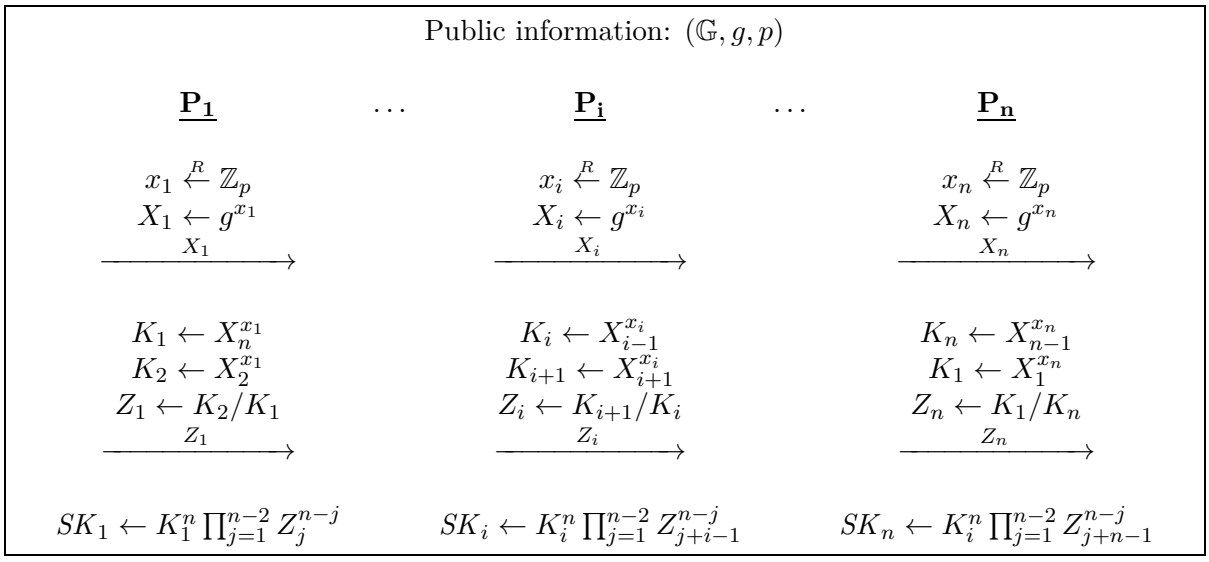


Figure 2.7: The Burmester-Desmedt group key exchange protocol [BD94, BD05]. The protocol works over a finite cyclic group $\mathbb{G}$ of prime order p generated by an element g.

As shown in [BD94, BD05], the Burmester-Desmedt GKE protocol is only secure against passive adversaries. In order to convert it into a GPAKE protocol, we make four main modifications to the Burmester-Desmedt GKE protocol. First, we add a first round of nonces to original protocol to clearly identify the different sessions. Second, we assume the existence of a family of random permutations $\mathcal{E}_k : \mathbb{G} \to \mathbb{G}$ indexed by a key k and use this family to encrypt the first round messages of the original Burmester-Desmedt GKE protocol. The encryption key k is derived via an ideal hash function (i.e., a random oracle) from the password, together with nonces, and the index of the user. Third, in order to avoid malleability attacks in which the adversary reorders messages in the same session or replay them in a different session, we add a final round of key confirmation. Finally, we change the key derivation step to make it depend on the shared group key as well as on the session identifier with the help of another ideal hash function. The resulting protocol is described in Figure 2.8.

As shown in Appendix C, the protocol in Figure 2.8 is a secure GPAKE protocol in the ideal-cipher and random-oracle models if the Decisional Diffie-Hellman (DDH) assumption described in Section 1.3 holds in $\mathbb{G}$.

Contributions. The GPAKE scheme in Figure 2.8 was the first *provably-secure* GPAKE scheme with a constant number of rounds. In Appendix C, in addition to proposing a new GPAKE protocol, we also provide concrete attacks against two constant-round GPAKE schemes previously proposed in the literature [DB06, LHL04].

2.7 GPAKE protocols in the standard model

As in the case of EKE-based protocols, the security of the GPAKE protocol in Figure 2.8 relies fundamentally on heuristic assumptions, due to the use of random oracles and ideal ciphers in the construction. To avoid having to rely on any idealized model, Pointcheval and I proposed a new GPAKE scheme in [AP06] (see Appendix D), which extends the Gennaro-Lindell (GL) PAKE protocol [GL03] to the group setting using ideas similar to those used in the Burmester-Desmedt GKE protocol [BD94, BD05]. Similarly to the GL PAKE protocol, the new protocol, to which we refer as AP, also relies on the notion of smooth projective hash functions. Hence, it

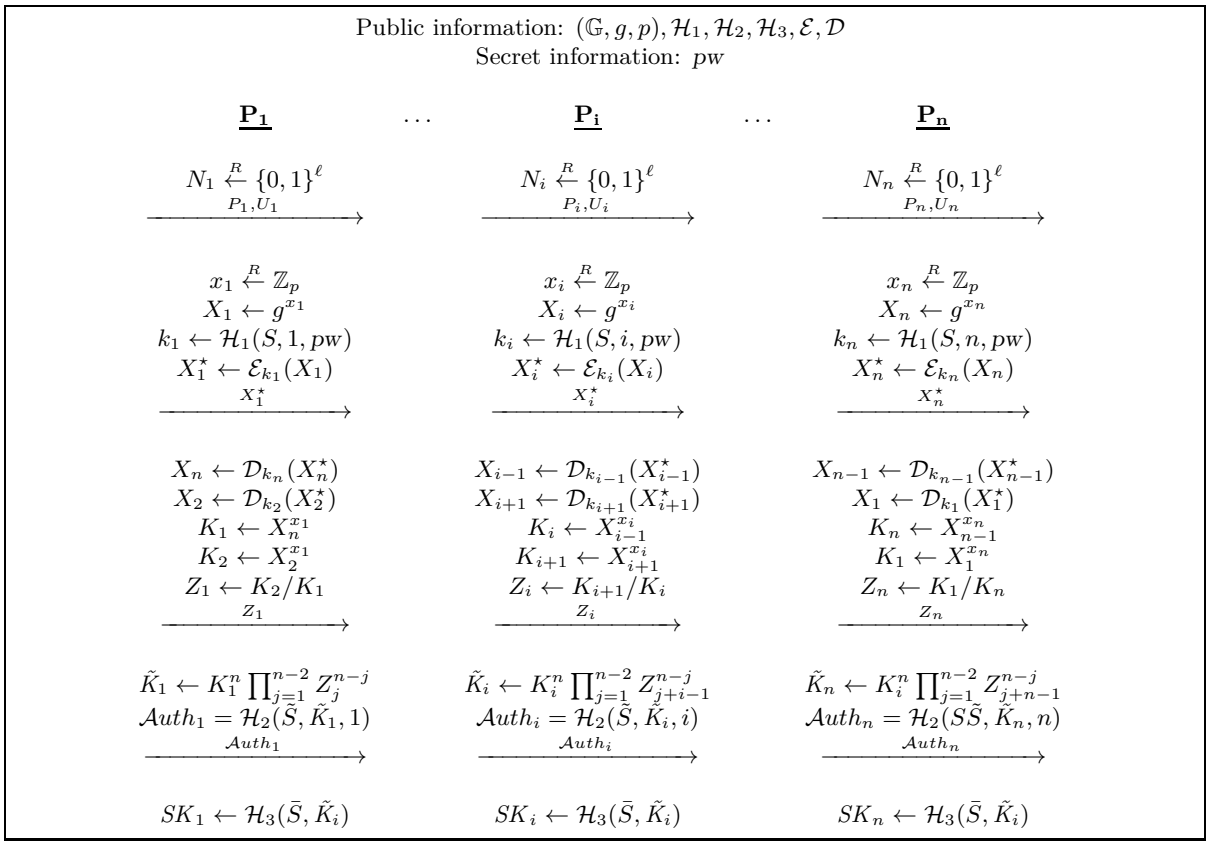


Figure 2.8: The password-authenticated group key exchange protocol in [ABCP06] (see Appendix C). The protocol works over a finite cyclic group $\mathbb{G}$ of prime order p generated by an element g. The elements S, $\tilde{S}$, and $\bar{S}$ are defined respectively as $P_1\|N_1\|\ldots\|P_n\|N_n$, $S\|X_1^\star\|Z_1\|\ldots\|X_n^\star\|Z_n$, and $\tilde{S}\|\mathcal{A}uth_1\|\ldots\|\mathcal{A}uth_n$. $\mathcal{E}$ and $\mathcal{D}$ are ideal symmetric encryption and decryption algorithms. $\mathcal{H}_1$, $\mathcal{H}_2$, and $\mathcal{H}_3$ are ideal hash functions. ℓ is a security parameter.

also enjoys efficient instantiations based on the decisional Diffie-Hellman, quadratic residuosity, and N-residuosity assumptions (see [GL03]).

Overview of the AP protocol [AP06]. The AP protocol is built in a modular way from four cryptographic primitives: a labeled encryption scheme, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Like the Burmester-Desmedt protocol, the AP protocol assumes a ring structure for the users and each user is associated with an index i between 1 and n, where n is the size of the group. After deciding on the order of the users, the protocol works as follows. First, each user in the group executes two correlated instances of the GL protocol, one with his predecessor and one with his successor so each user can authenticate his neighbors (this accounts for the first 3 rounds of the protocol). However, instead of generating a single session key in each of these instances, the original GL protocol is modified so that two independent session keys are generated in each session. Next, the first of these keys is used as a test key to authenticate the neighbor with whom that key is being shared and the other one is used to help the computation of the group session key, which is defined as the product of these latter keys. As in the Burmester-Desmedt protocol, this is achieved via an additional round of communication in which each user computes and broadcasts the ratio of the session keys that he shares with his predecessor and successor. After this round, each user is capable of computing the group session key. Still, to ensure that all users agree on the same key, a final round of signatures is added to the protocol to make sure that all users compute

the group session key based on the same transcript. The key used to verify the signature of a user is the same one transmitted by that user in the first round of the GL protocol. A pictorial description can be found in Figure 2.9.

Security of the AP protocol. Let PKE be a labeled encryption secure against chosen-ciphertext attacks and let HASH be a family of smooth projective hash functions for the language consisting of triples $\{(c, \ell, pw)\}$ such that c is an encryption of the password pw with label ℓ. Let UH and UH' be two families of universal hash functions, and let SIG be a signature scheme that is strongly unforgeable against chosen-message attacks. Then, as shown in Appendix D, the protocol in Figure 2.9 is a secure GPAKE protocol in the standard model according to the definition in Section 2.2.1.

2.8 GPAKE protocols in the UC model

Like most existing GPAKE protocols, the GPAKE protocols that we discussed so far have proofs in the indistinguishability-based security model discussed in Section 2.2.1, which does not guarantee security under arbitrary composition. To address this problem, together with Catalano, Chevalier, and Pointcheval, we defined in [ACCP08] an ideal functionality for GPAKE with explicit authentication and contributiveness in the UC framework. As with previous definitions in the same framework, our definitions do not assume any particular distribution on passwords or independence between passwords of different parties. In addition to the new definition, we also provided the first steps towards realizing this functionality by analyzing a variant of the GPAKE protocol in Figure 2.8 and showing that it realizes the new ideal functionality in the random-oracle and ideal-cipher models based on the computational Diffie-Hellman (CDH) assumption described in Section 1.3.

To overcome the limitations of the protocol in [ACCP08], Chevalier, Granboulan, Pointcheval, and I proposed in [ACGP11] a new generic construction of password-authenticated group key exchange protocol from any UC-secure PAKE with explicit authentication. The new construction has several advantages when compared to existing solutions. First, it only assumes a common reference string and does not rely on any idealized model. Second, it enjoys a simple and intuitive security proof in the UC framework and is optimal in the sense that it allows at most one password test per user instance. Third, it also achieves a strong notion of security against insiders in that the adversary cannot bias the distribution of the session key as long as one of the players involved in the protocol is honest. Finally, the new construction can be easily extended to the dynamic case in a way that the costs of establishing a common key between two existing groups is significantly smaller than computing a common key from scratch.

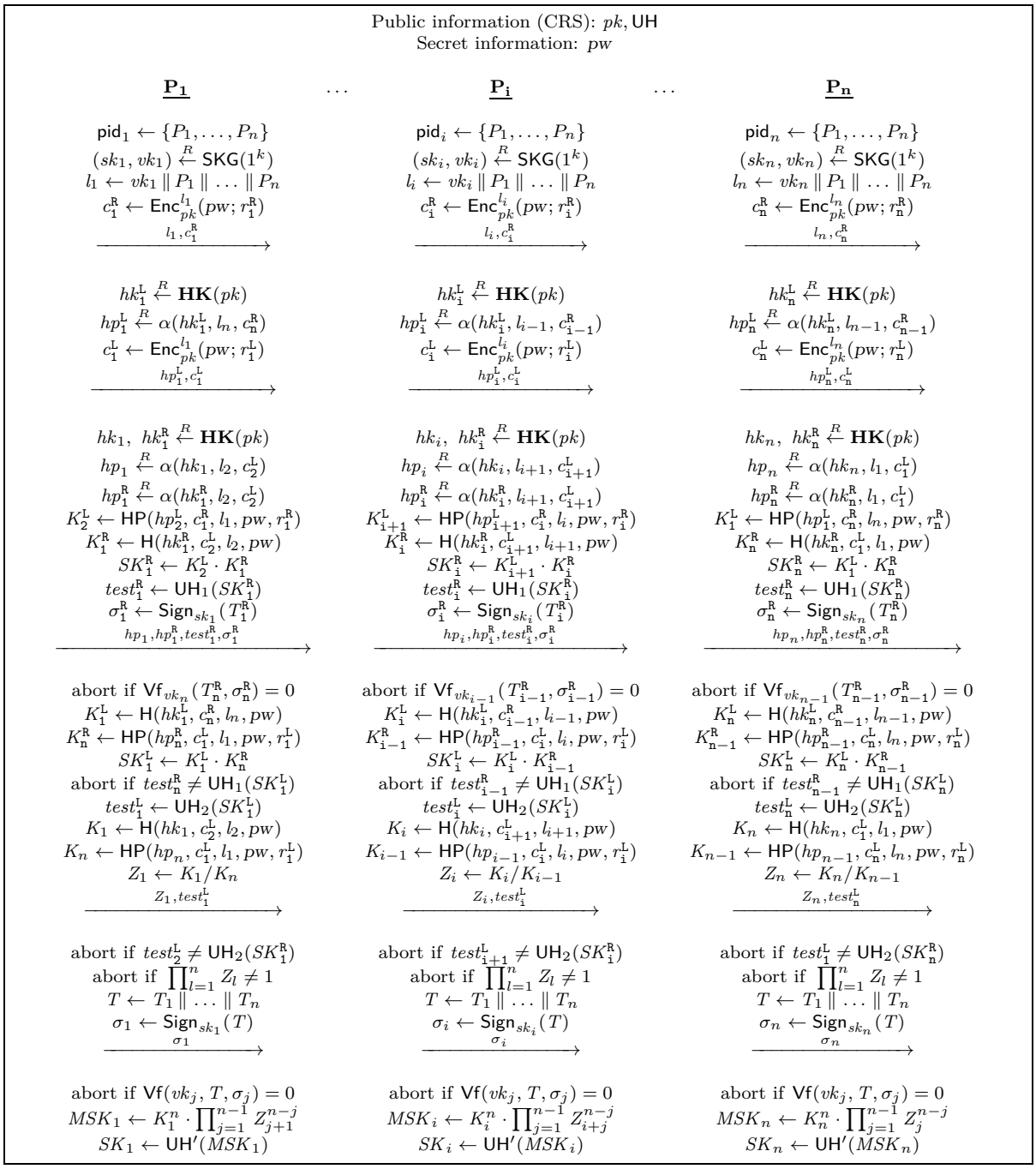


Figure 2.9: The GPAKE protocol in [AP06] (also in Appendix D) with n players $\{P_1, \ldots, P_n\}$, where $T_{\mathtt{i}}^{\mathtt{R}}$ and T_i are defined respectively as $P_i \,\|\, P_{i+1} \,\|\, c_{\mathtt{i}}^{\mathtt{R}} \,\|\, c_{\mathtt{i+1}}^{\mathtt{L}} \,\|\, hp_i \,\|\, hp_{\mathtt{i}}^{\mathtt{R}} \,\|\, hp_{\mathtt{i+1}}^{\mathtt{L}} \,\|\, test_{\mathtt{i}}^{\mathtt{R}}$ and $vk_i \,\|\, P_i \,\|\, c_i \,\|\, hp_i \,\|\, hp_{\mathtt{i}}^{\mathtt{L}} \,\|\, hp_{\mathtt{i}}^{\mathtt{R}} \,\|\, Z_i \,\|\, SK_{\mathtt{i}}^{\mathtt{L}}$ for $i = 1, \ldots, n$. $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are the key generation, encryption, and decryption algorithms of a labeled public-key encryption scheme [Sho04]. $(\mathsf{SKG}, \mathsf{Sign}, \mathsf{Vf})$ are the key generation, signing, and verification algorithms of a signature scheme. $(\mathsf{HK}, \alpha, \mathsf{H}, \mathsf{HP})$ are the key generation, key projection, hashing, and projected hashing algorithms of a family of smooth projective hash functions for the language L consisting of triples $\{(c, \ell, pw)\}$ such that c is an encryption of the password pw with label ℓ. UH and UH' are two universal hash functions chosen uniformly at random from their respective families [HILL99], with $\mathsf{UH}_1(\cdot)$ and $\mathsf{UH}_2(\cdot)$ referring to the first and second halves of $\mathsf{UH}(\cdot)$.

Chapter 3

Identity-based cryptography

3.1 Introduction

In order to link users to their public keys, public-key cryptosystems have to rely on the existence of a public-key infrastructure (PKI), where a trusted authority certifies the relation between users and their public keys by means of a digital signature. Since the costs of maintaining such a PKI can be prohibitive, Shamir proposed the concept of identity-based cryptography in [Sha85], which is a generalization of the standard notion of public-key cryptography. In identity-based cryptography, the public key of a user is his identity (e.g., his name or email address) and the corresponding private key is handed to him by a trusted key distribution center. In the particular case of identity-based encryption schemes, given an email address and a set of user-independent public parameters, one can encrypt a message to the owner of the email address without needing to obtain an authentic copy of the owner's public key first.

Since being introduced by Shamir in 1984 [Sha85], identity-based cryptography has received a lot of attention due to the fact that one no longer needs to maintain a separate public key for each user. While an efficient construction of identity-based signatures was also proposed in the same paper, it was only in 2001 that the first practical identity-based encryption (IBE) construction appeared in the literature based on elliptic-curve pairings [BF03]. Later that year, Cocks proposed an alternative IBE construction based on the quadratic residuosity problem [Coc01].

Overview of this chapter. Since most of my work in the area of identity-based cryptography concerns the design of encryption schemes, the remainder of this chapter will focus on the latter. In Section 3.2, we start by recalling some standard definitions and security notions for IBE schemes that we will be using in the following sections. This is followed by a review of some of the main constructions in the identity-based setting in Section 3.3, such as the seminal IBE scheme by Boneh and Franklin [BF03] and the hierarchical IBE scheme by Boneh and Boyen [BB04a]. After reviewing standard IBE constructions, we discuss in the remaining sections generalizations and applications of the IBE schemes. First, in Section 3.4, we present a generalization of IBE in Section 3.4, known as IBE with wildcards, in which users can simultaneously encrypt a message to a group of users matching a certain pattern defined through a sequence of fixed strings and wildcards. Then, in Section 3.5, we discuss an extension of public-key encryption, known as public-key encryption with keyword search (PEKS), which is closely related to IBE schemes. As we show in Section 3.5, A PEKS scheme can be seen as a special form of IBE in which a user's decryption key only allows him to test whether he is the intended target of a ciphertext or not. Finally, in Section 3.6, we discuss some further generalizations.

3.2 Security notions for identity-based encryption

In this section, we review some of the standard security notions used in the context of the hierarchical identity-based encryption (HIBE), such as data privacy, anonymity, and robustness. Towards this goal, we first recall their syntax.

3.2.1 Syntax

The concept of *identity-based encryption* (IBE) is a generalization of the standard notion of public-key encryption in which the sender can encrypt messages to a user based only on the identity of the latter and a set of user-independent public parameters. In these systems, there exists a trusted authority, called private key generator, that is responsible for generating decryption keys for all identities in the system.

More formally, an IBE scheme is defined by a tuple of algorithms $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$, a message space $\mathcal{M}$ and an identity space ID. The scheme provides the following functionality. The algorithm Setup is run by a trusted authority to generate a pair of keys (mpk, msk) such that mpk is made public, whereas msk is kept private. On input an identity $id \in \mathsf{ID}$, the key derivation algorithm $\mathsf{KeyDer}(msk, id)$ uses the master secret key to generate a key sk_{id} for the user with identity id. Then, every user holding the master public key mpk, can encrypt a message $m \in \mathcal{M}$ for the identity id by running $C \xleftarrow{R} \mathsf{Enc}(mpk, id, m)$. Finally, the ciphertext C can be decrypted by running the deterministic decryption algorithm, $m \leftarrow \mathsf{Dec}(sk_{id}, C)$. For correctness, it is required that for all honestly generated master keys $(mpk, msk) \xleftarrow{R} \mathsf{Setup}$, for all messages $m \in \mathcal{M}$ and all identities $id \in \mathsf{ID}$, $m \leftarrow \mathsf{Dec}(\mathsf{KeyDer}(msk, id), \mathsf{Enc}(mpk, id, m))$ holds with all but negligible probability.

Soon after being proposed, the notion of IBE was generalized to the hierarchical setting by Horwitz and Lynn [HL02], who considered the scenario in which intermediate nodes can act as private key generators. In a hierarchical identity-based encryption (HIBE) scheme, users are hierarchically organized in a tree of depth L whose root is the trusted authority. The identity of a user at level $1 \leq \ell \leq L$ is represented by a vector $id = (id_1, \ldots, id_\ell) \in \mathsf{ID}^\ell$. Similarly to IBE, a HIBE scheme can be defined more formally by a tuple of algorithms $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ working as an IBE except that a user at level ℓ with identity $id = (id_1, \ldots, id_\ell)$ can use the key derivation algorithm $\mathsf{KeyDer}(sk_{id}, id')$ to generate a secret key for any of its children $id' = (id_1, \ldots, id_\ell, id_{\ell+1})$ at level $\ell + 1$. Since this process can be iterated, every user can generate keys for all its descendants. In a HIBE, the encryption algorithm takes as input a vector of identities. The correctness requirement is almost the same as that of IBE, except that it is extended so that a ciphertext $C \xleftarrow{R} \mathsf{Enc}(mpk, id, m)$ can be decrypted by any identity id' which is an ancestor of id. Finally, note that an IBE is a particular case of a HIBE with $L = 1$.

3.2.2 Data privacy

The now-standard definition of security of (H)IBE schemes, first suggested by Boneh and Franklin [BF03], is indistinguishability under *adaptive-identity* chosen-plaintext attacks (IND-CPA). In this security model, the adversary is allowed to obtain secret keys for adaptively chosen identities before deciding the identity and a pair of messages upon which it wishes to be challenged. By allowing these queries, this notion implicitly captures resistance against collusion attacks as different users should be unable to combine their keys in an attempt to decrypt ciphertexts intended to another user.

To define this security notion more precisely, we consider the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ described in Figure 3.1 using the notation of code-based games. The game is defined by four procedures and is executed with an adversary $\mathcal{A}$ as follows. The procedure **Initialize** initializes the set U

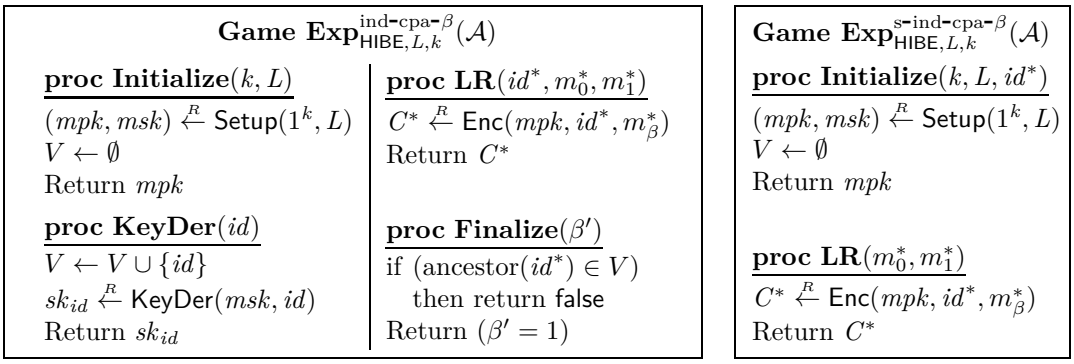


Figure 3.1: Games $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{s-ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ defining IND-CPA and s-IND-CPA security (respectively) of a hierarchical identity-based encryption scheme $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ of depth L. The procedures **KeyDer** and **Finalize** are common to both games, which differ in their **Initialize** and **LR** procedures.

and the decryption key table DK to empty, generates a fresh key pair $(mpk, msk) \xleftarrow{R} \mathsf{Setup}$ and returns mpk to the adversary. During the execution of the game, the adversary is allowed to make queries to the **KeyDer** procedure that, on input an identity $id = (id_1, \ldots, id_\ell)$, returns the secret key $usk_{id} \xleftarrow{R} \mathsf{KeyDer}(msk, id)$ corresponding to identity id. $\mathcal{A}$ is also allowed to make a single query (id^*, m_0^*, m_1^*) to the **LR** procedure, where $m_0^*, m_1^* \in \{0,1\}^*$ are assumed to have the same length. To answer it, the game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ generates a challenge ciphertext $C^* \xleftarrow{R} \mathsf{Enc}(mpk, id^*, m_\beta^*)$ and gives C^* to $\mathcal{A}$. Eventually, the adversary ends the game by querying the **Finalize** procedure with a guess β' for the bit β used to generate the challenge ciphertext.

The advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ is then defined as the probability that game $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ outputs true minus the probability that game $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ outputs true. Note that game $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ returns true only if $\mathcal{A}$ outputs 1 without ever having queried the key derivation oracle on any ancestor identity $id = (id_1^*, \ldots, id_\ell^*)$ of id^*, $\ell \leq \ell^*$. Finally, a HIBE scheme HIBE is said to be secure if $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ is a negligible function in k for all PTAs $\mathcal{A}$.

A weaker definition of data privacy for (H)IBE schemes, suggested by Canetti, Halevi, and Katz in [CHK03], is indistinguishability under *selective-identity* chosen-plaintext attacks (s-IND-CPA). Unlike the standard IND-CPA definition, the adversary in the s-IND-CPA security game has to commit to the challenge identity before obtaining the public parameters of the scheme. As shown in [CHK03], selective-identity security may be sufficient for certain applications, such as the constructions of forward-secure encryption schemes. The experiment describing the s-IND-CPA security is also described in Figure 3.1.

3.2.3 Anonymity

In addition to data privacy, which is captured by the notion of indistinguishability under adaptive-identity chosen-plaintext attacks, another useful security notion is *anonymity* [ABC+08, ABC+05a, BBDP01]. Informally, anonymity under *adaptive-identity* chosen-plaintext attacks (ANO-CPA) asks that a ciphertext does not reveal the identity under which it was created, even when the adversary is allowed to obtain secret keys for adaptively chosen identities of its choice. Though originally considered in the context of public-key encryption schemes in [BBDP01], this notion has found several other applications in the identity-based setting (see Appendix F). In particular, as we show in Section 3.5, anonymity is extremely important when constructing

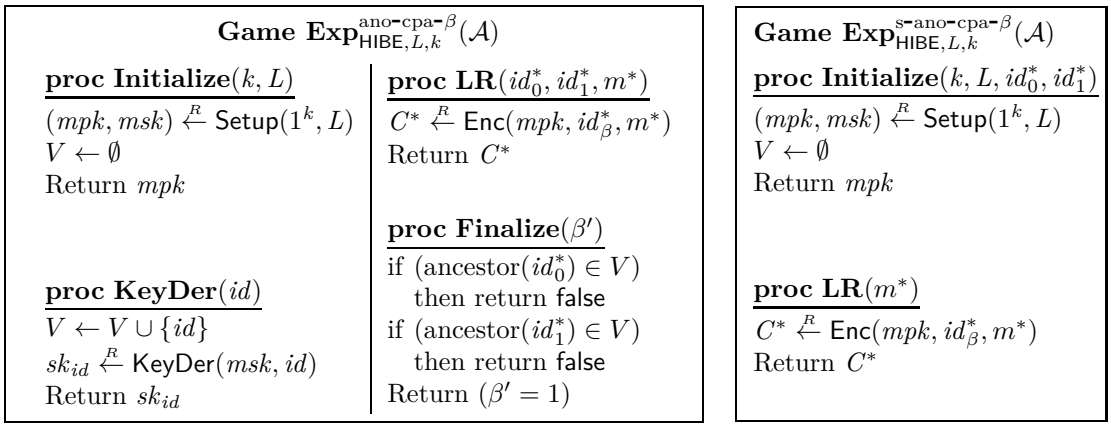


Figure 3.2: Games $\mathbf{Exp}^{\text{ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{s-ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ defining ANO-CPA and s-ANO-CPA security (respectively) of a hierarchical identity-based encryption scheme $\mathsf{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ of depth L. The procedures **KeyDer** and **Finalize** are common to both games, which differ in their **Initialize** and **LR** procedures.

public-key encryption schemes with keyword search [BDOP04] from IBE schemes. As in the IND-CPA case, the anonymity notion ANO-CPA implicitly captures resistance against collusion attacks by allowing the adversary to query the key derivation oracle adaptively, as different users should be unable to combine their keys in an attempt to decrypt ciphertexts intended to another user.

As described in Appendix F, the notion of anonymity can be defined more formally by considering the game $\mathbf{Exp}^{\text{ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ described in Figure 3.2 using the notation of code-based games. As in the IND-CPA case, the $\mathbf{Exp}^{\text{ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ game is defined by four procedures, which are executed with an adversary $\mathcal{A}$. The procedures **Initialize**, **KeyDer**, and **Finalize** are defined as in the $\mathbf{Exp}^{\text{ind-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ game. During the execution of the game, the adversary $\mathcal{A}$ is allowed to make a single query (id^*, m_0^*, m_1^*) to the **LR** procedure, where $m_0^*, m_1^* \in \{0,1\}^*$ and id_0^* and id_1^* have length at most L. To answer it, the game generates a challenge ciphertext $C^* \xleftarrow{R} \mathsf{Enc}(mpk, id_\beta^*, m^*)$ and returns C^* to $\mathcal{A}$. The game ends when the adversary queries the **Finalize** procedure with a guess β', which is considered the output of the game $\mathbf{Exp}^{\text{ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$.

The advantage $\mathbf{Adv}^{\text{ano-cpa}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ is then defined as the probability that game $\mathbf{Exp}^{\text{ano-cpa-1}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ returns true minus the probability that $\mathbf{Exp}^{\text{ano-cpa-0}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ returns true. Note that game $\mathbf{Exp}^{\text{ano-cpa-}\beta}_{\mathsf{HIBE},L,k}(\mathcal{A})$ returns true only if $\mathcal{A}$ outputs 1 without ever having queried the key derivation oracle on any ancestor identity id of the challenge identities id_0^* and id_1^*. Finally, a HIBE scheme HIBE is said to be anonymous if $\mathbf{Adv}^{\text{ano-cpa}}_{\mathsf{HIBE},L,k}(\mathcal{A})$ is a negligible function in k for all PTAs $\mathcal{A}$.

For completeness, the notion of anonymity under *selective-identity* chosen-plaintext attacks (s-ANO-CPA), in which the adversary has to commit to the challenge identities before obtaining the public parameters of the scheme is also described in Figure 3.2.

Contributions. Besides formalizing the notion of anonymity in the identity-based setting in [ABC+05a, ABC+08] (see Appendix F), we also proved that the IBE scheme due to Boneh and Franklin in [BF03] is anonymous in the random-oracle model if the BDH problem described in Section 1.3 is hard in the underlying group. Moreover, we also proved that a modified version of the Gentry-Silverberg HIBE scheme in [GS02] achieves a limited form of anonymity under similar assumptions.

3.2.4 Robustness

The notion of robustness, introduced by myself together with Bellare and Neven in [ABN10] (see Appendix G), reflects the difficulty of producing a ciphertext which is valid under two different encryption keys. More precisely, suppose C is an IBE ciphertext for an encrypted message M under an identity id_0. We know that if C is decrypted using the secret key sk_0 corresponding to id_0, the result would be M. But what if we decrypt C using a secret key sk_1 corresponding to an identity $id_1 \neq id_0$? Previous security notions for identity-based encryption are silent about this. Roughly, we refer to a scheme as "robust" if the result of this decryption is $\perp$, meaning that the decryption algorithm rejects.

Robustness can be trivially achieved by appending the public key or identity of the intended recipient to the ciphertext, and by checking for it upon decryption. However, this solution comes at the expense of anonymity and is only acceptable in cases where the latter does not need to be preserved. Unfortunately, finding a solution which achieves robustness while preserving anonymity is not so easy. As shown in Appendix G, natural (anonymity-preserving) solutions, such as including the identity of the recipient in the plaintext and have recipients check this upon decryption, do not work in general as there are (anonymous) IBE schemes where it is possible for an adversary to create a ciphertext that multiple recipients will accept. This is where the notion of robustness comes into play. If the encryption scheme is robust, then only the intended recipient will obtain a valid result upon decryption, meaning one different from $\perp$. In hindsight, the natural solutions we just mentioned are just attempts to add robustness without violating anonymity.

Two different flavors of robustness are defined in Appendix G, depending on whether ciphertexts are honestly or adversarially generated. In a *weakly* robust IBE scheme, the adversary outputs a pair (id_0, id_1) of distinct identities together with a message m^* and is considered successful if the decryption of an honest encryption of the message m^* for identity id_0 under the decryption key sk_1 corresponding to id_1 returns non-$\perp$. In a *strongly* robust IBE scheme, the adversary outputs a pair (id_0, id_1) of distinct identities together with a ciphertext C^* and is considered successful if the decryption of C^* returns non-$\perp$ under both decryption keys sk_0 and sk_1 corresponding to identities id_0 and id_1. Both weak and strong robustness can be considered under chosen-plaintext (CPA) or chosen-ciphertext (CCA) attacks, depending on whether the adversary is given access to a decryption oracle. The resulting four notions are denoted WROB-CPA, WROB-CCA, SROB-CPA, and SROB-CCA.

As shown in Appendix G, the notions of weak and strong robustness can be defined more formally by considering the games $\mathbf{Exp}^{\mathrm{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ and $\mathbf{Exp}^{\mathrm{srob}}_{\mathsf{IBE},k}(\mathcal{A})$ described in Figure 3.3 using the notation of code-based games. Both games are defined by four procedures, which are executed with an adversary $\mathcal{A}$. The procedures **Initialize**, **KeyDer** and **Dec** are common to both games, which differ in their **Finalize** procedures. While the adversary $\mathcal{A}$ makes no **Dec** queries in a chosen-plaintext attack (CPA), it might ask such queries in a chosen-ciphertext attack (CCA). The procedure **Initialize** initializes the set V to empty, generates a fresh key pair $(mpk, msk) \xleftarrow{R} \mathsf{Setup}$, and returns mpk to the adversary. To answer a **KeyDer** query id made by $\mathcal{A}$, the game returns the secret key $usk_{id} \xleftarrow{R} \mathsf{KeyDer}(msk, id)$ corresponding to identity id. To answer a **Dec** query (C, id), the game generates a secret key usk_{id} for id if necessary, computes $m \leftarrow \mathsf{Dec}(usk_{id}, C)$, and returns m to the adversary. Finally, in the game $\mathbf{Exp}^{\mathrm{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$, the adversary outputs a challenge message $m^* \in \{0,1\}^*$ and two challenge identities id_0^* and id_1^* and wins the game if the encryption of m^* under identity id_0^* decrypts correctly under the secret key corresponding to identity id_1^*. In the game $\mathbf{Exp}^{\mathrm{srob}}_{\mathsf{IBE},k}(\mathcal{A})$, the adversary outputs a challenge ciphertext C^* and two challenge identities id_0^* and id_1^* and wins the game if C^* decrypts correctly under the secret keys corresponding to id_0^* and id_1^*.

The advantage $\mathbf{Adv}^{\mathrm{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ (resp. $\mathbf{Adv}^{\mathrm{srob}}_{\mathsf{IBE},k}(\mathcal{A})$) is then defined as the probability that

Games $\mathbf{Exp}^{\text{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{srob}}_{\mathsf{IBE},k}(\mathcal{A})$

proc Initialize(k)	**proc Finalize**(id_0^*, id_1^*, m^*) // WROB
$(pk, msk) \xleftarrow{R} \mathsf{Setup}(1^k)$	if $(id_0^* \in V) \vee (id_1^* \in V)$ then return false
$V \leftarrow \emptyset$	if $(id_0^* = id_1^*)$ then return false
Return mpk	$m_0 \leftarrow m^*$; $C \xleftarrow{R} \mathsf{Enc}(pk, id_0, m_0)$
	$sk_1 \xleftarrow{R} \mathsf{KeyDer}(msk, id_1)$
	$m_1 \leftarrow \mathsf{Dec}(sk_1, C^*)$
proc KeyDer(id)	Return $(m_0 \neq \perp) \wedge (m_1 \neq \perp)$
$V \leftarrow V \cup \{id\}$	
$sk_{id} \xleftarrow{R} \mathsf{KeyDer}(msk, id)$	**proc Finalize**(id_0^*, id_1^*, C^*) // SROB
Return sk_{id}	if $(id_0^* \in V) \vee (id_1^* \in V)$ then return false
	if $(id_0^* = id_1^*)$ then return false
	$sk_0 \xleftarrow{R} \mathsf{KeyDer}(msk, id_0^*)$
proc Dec(C, id)	$sk_1 \xleftarrow{R} \mathsf{KeyDer}(msk, id_1^*)$
$sk_{id} \xleftarrow{R} \mathsf{KeyDer}(msk, id)$	$m_0 \leftarrow \mathsf{Dec}(pk, id_0^*, sk_0, C^*)$
$m \leftarrow \mathsf{Dec}(sk_{id}, C)$	$m_1 \leftarrow \mathsf{Dec}(pk, id_1^*, sk_1, C^*)$
Return m	Return $(m_0 \neq \perp) \wedge (m_1 \neq \perp)$

Figure 3.3: Games $\mathbf{Exp}^{\text{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ and $\mathbf{Exp}^{\text{srob}}_{\mathsf{IBE},k}(\mathcal{A})$ defining WROB-ATK and SROB-ATK security (respectively) of an identity-based encryption scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$. The procedures **Initialize**, **KeyDer** and **Dec** are common to both games, which differ in their **Finalize** procedures. ATK $\in \{\text{CPA}, \text{CCA}\}$ indicates whether the adversary is given access to the **Dec** procedure.

game $\mathbf{Exp}^{\text{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ (resp. $\mathbf{Exp}^{\text{srob}}_{\mathsf{IBE},k}(\mathcal{A})$) returns true. Finally, a IBE scheme IBE is said to be weakly (resp. strongly) robust if $\mathbf{Adv}^{\text{wrob}}_{\mathsf{IBE},k}(\mathcal{A})$ (resp. $\mathbf{Adv}^{\text{srob}}_{\mathsf{IBE},k}(\mathcal{A})$) is a negligible function in k for all PTAs $\mathcal{A}$.

Contributions. In addition to introducing the notion of robustness, we consider and dismiss natural approaches to achieve it in Appendix G. We also provide two general robustness-adding transforms; test robustness of existing schemes and patch the ones that fail; and discuss some applications.

3.3 Identity-based encryption constructions

The Boneh-Franklin IBE scheme. The first practical IBE construction to appear in the literature was due to Boneh and Franklin [BF03], based on elliptic-curve pairings. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\text{T}}$ be a non-degenerate bilinear map as defined in Section 1.2.4, let g be a generator of $\mathbb{G}$, and let $H_1 : \{0,1\}^* \to \mathbb{G}^*$ and $H_2 : \mathbb{G}_{\text{T}} \to \{0,1\}^k$ be random oracles. In the basic version of Boneh-Franklin IBE scheme, called $\mathsf{BasicIdent}$ and described in Figure 3.4, the master secret key is an exponent $s \xleftarrow{R} \mathbb{Z}_p^*$ and the corresponding master public key is $S \leftarrow g^s$. Then, given S and the user's identity id, the user's ephemeral public key g_{id} can be computed as the pairing between S and the hash of id, i.e. $g_{id} \leftarrow \hat{e}(S, H_1(id))$. To encrypt a message m under identity id, the sender simply chooses $r \xleftarrow{R} \mathbb{Z}_p^*$ and outputs the tuple $(g^r, m \oplus H_2(g_{id}^r))$ as the ciphertext. To decrypt a ciphertext (C_1, C_2), the user with identity id first needs to obtain a decryption key $usk \leftarrow H_1(id)^s$ from the master authority and then compute $m \leftarrow C_2 \oplus H_2(\hat{e}(C_1, usk))$.

Security of $\mathsf{BasicIdent}$. To understand why the $\mathsf{BasicIdent}$ IBE scheme is correct, note that $\hat{e}(C_1, usk) = \hat{e}(g^r, H_1(id)^s) = \hat{e}(g^s, H_1(id))^r = g_{id}^r$. To understand informally why it is also secure, first note that ephemeral public values g_{id} used in the encryption of a message for user

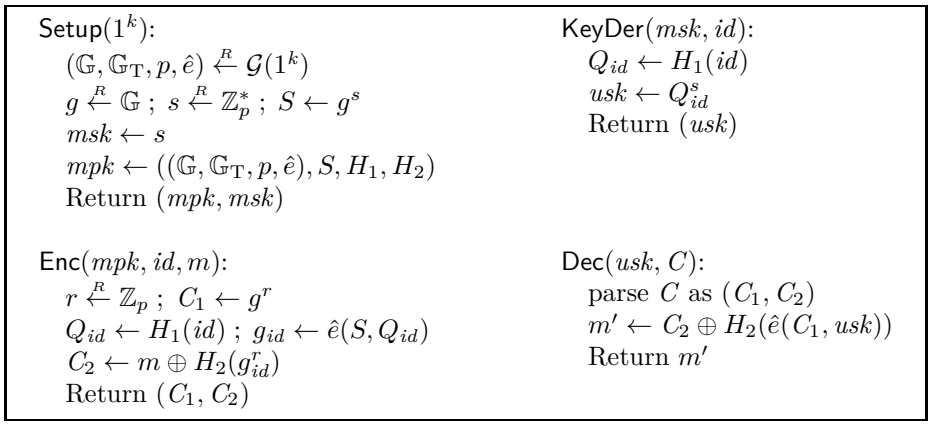


Figure 3.4: The Boneh-Franklin BasicIdent IBE scheme [BF03], where $\mathcal{G}$ is a pairing parameter generator and $H_1 : \{0,1\}^* \to \mathbb{G}^*$ and $H_2 : \mathbb{G}_T \to \{0,1\}^k$ are random oracles.

id are independent of each other due to the use of the random oracle H_1 in its computation. Likewise, the use of H_1 in the key derivation algorithm also protects against user collusion by making the decryption key values *usk* independent of each other. Finally, let (m_0^*, m_1^*) and id^* be the challenge message pair and identity and let (C_1^*, C_2^*) be the challenge ciphertext, where $C_1^* = g^{r^*}$ and $C_2^* = m_\beta^* \oplus H_2(\hat{e}(S, H_1(id^*))^{r^*})$ and β is the hidden challenge bit. Since the values S, C_1^*, $H_1(id^*)$ look random to the adversary, the latter can only distinguish the encryption of m_0^* from that of m_1^* if it queries the random oracle H_2 on input $\hat{e}(S, H_1(id))^{r^*}$. However, since this amounts to breaking the BDH problem, no PTA adversary should succeed in this task with non-negligible probability. As shown formally in [BF03], this is indeed the case.

Hierarchical setting. In order to consider scenarios in which intermediate nodes can act as private key generators, Horwitz and Lynn proposed in [HL02] a generalization of IBE known as hierarchical identity-based encryption (HIBE). In their paper, they also provided a two-level HIBE construction based on the Boneh-Franklin IBE scheme, but their scheme could provide full collusion resistance only in the upper level. The first HIBE scheme to provide full collusion resistance in all levels is due to Gentry and Silverberg [GS02]. Like the Horwitz-Lynn HIBE scheme, the Gentry-Silverberg HIBE scheme was also based on the Boneh-Franklin IBE scheme and proven secure in the random-oracle model [BR93].

The first HIBE to be proven secure in the standard model is due to Canetti, Halevi, and Katz [CHK03], but in a weaker security model, called the *selective-identity* model. Unlike the IND-CPA security definition, the selective-identity model requires the adversary to commit to the challenge identity before obtaining the public parameters of the scheme (see Figure 3.1 in Section 3.2.2). Despite providing weaker security guarantees, Canetti, Halevi, and Katz showed that the selective-identity model is sufficient for building forward-secure encryption schemes, which was the main motivation of their paper.

The work by Canetti, Halevi, and Katz was soon improved by Boneh and Boyen [BB04a], who provided two very efficient selective-identity-secure (H)IBE constructions in the standard model: one based on the BDDH assumption described in Section 1.3 and another one based on the Decision Bilinear Diffie-Hellman Inversion (Decision BDHI) assumption, which was introduced in the same paper. Since the first of these HIBE schemes became the basis of several other schemes in the literature, let us recall it here.

The BB-HIBE *scheme.* Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate bilinear map as defined in Section 1.2.4 and let g_1 and g_2 be two random generators of $\mathbb{G}$. In the first Boneh-Boyen HIBE scheme [BB04a] (BB-HIBE), which is described in Figure 3.5, identities are assumed to

be vectors of elements of $\mathbb{Z}_p^*$. If necessary, this can be achieved by applying a collision-resistant hash function $h : \{0,1\}^* \to \mathbb{Z}_p^*$ to binary identities before applying the scheme. To generate the master public key, the master authority first chooses $\alpha \xleftarrow{R} \mathbb{Z}_p$, then computes $h_1 \leftarrow g_1^\alpha$, $h_2 \leftarrow g_2^\alpha$, and $u_i \xleftarrow{R} \mathbb{G}$ for $i = 0 \ldots L$, and sets $mpk \leftarrow (g_1, g_2, h_1, u_0, \ldots, u_L)$ and $msk \leftarrow h_2$. To encrypt a message m for user $id = (id_1, \ldots, id_\ell)$, the sender simply chooses $r \xleftarrow{R} \mathbb{Z}_p^*$, and outputs the tuple $(g_1^r, ((u_1 \cdot u_0^{id_1})^r, \ldots, (u_\ell \cdot u_0^{id_\ell})^r), m \cdot \hat{e}(h_1, g_2)^r)$ as the ciphertext. To decrypt a ciphertext (C_1, $C_{2,1}, \ldots, C_{2,\ell}, C_3$), the user with identity $id = (id_1, \ldots, id_\ell)$ first obtains a decryption key (usk_0, $usk_1, \ldots, usk_\ell$) $= (h_2 \prod_{i=1}^{\ell} (u_i \cdot u_0^{id_i})^{r_i}, g_1^{r_1}, \ldots, g_1^{r_\ell})$ from the master authority and then computes $m \leftarrow C_3 \cdot \prod_{i=1}^{\ell} \hat{e}(usk_i, C_{2,i}) / \hat{e}(C_1, usk_0)$.

$\mathsf{Setup}(1^k, L)$:
- $(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \xleftarrow{R} \mathcal{G}(1^k)$
- $g_1, g_2 \xleftarrow{R} \mathbb{G}$; $\alpha \xleftarrow{R} \mathbb{Z}_p$
- $h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
- $u_i \xleftarrow{R} \mathbb{G}$ for $i = 0 \ldots L$
- $mpk \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g_1, g_2, h_1, u_0, \ldots, u_L)$
- $msk \leftarrow h_2$
- Return (mpk, msk)

$\mathsf{Enc}(mpk, id, m)$:
- parse id as $(id_1, \ldots, id_\ell)$
- $r \xleftarrow{R} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
- for $i = 1, \ldots, \ell$ do
  - $C_{2,i} \leftarrow \left(u_i \cdot u_0^{id_i}\right)^r$
- $C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
- Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

$\mathsf{KeyDer}(usk_{(id_1,\ldots,id_\ell)}, id_{\ell+1})$:
- parse $usk_{(id_1,\ldots,id_\ell)}$ as $(usk_0, \ldots, usk_\ell)$
- $r_{\ell+1} \xleftarrow{R} \mathbb{Z}_p$
- $usk'_0 \leftarrow usk_0 \cdot \left(u_{\ell+1} \cdot u_0^{id_{\ell+1}}\right)^{r_{\ell+1}}$
- $usk'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
- Return $(usk'_0, usk_1, \ldots, usk_\ell, usk'_{\ell+1})$

$\mathsf{Dec}(usk_{(id_1,\ldots,id_\ell)}, C)$:
- parse $usk_{(id_1,\ldots,id_l)}$ as $(usk_0, \ldots, usk_\ell)$
- parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
- $m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(usk_i, C_{2,i})}{\hat{e}(C_1, usk_0)}$
- Return m'

Figure 3.5: The BB-HIBE scheme [BB04a], where $\mathcal{G}$ is a pairing parameter generator. The description of KeyDer assumes $usk_\varepsilon = (usk_0) = (msk) = (h_2)$.

Security of BB-HIBE. As shown in [BB04a], the BB-HIBE scheme in Figure 3.5 is a selective-identity-secure HIBE scheme if the BDDH problem described in Section 1.3 is hard with respect to the parameters $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ generated by pairing parameter generator $\mathcal{G}$. In the same paper, the authors also showed a generic transformation in the random oracle model that converts any selective-identity-secure (H)IBE scheme into a (H)IBE that is secure against adaptive-identity chosen-plaintext attacks, described in Section 3.2.2.

3.4 Identity-based encryption with wildcards

One of the main motivating applications of (H)IBE is email-based encryption where the identity of a user is his email address. In these applications, one can encrypt a message to the owner of the email address without needing to obtain an authentic copy of the owner's public key first. Motivated by the fact that many email addresses correspond to groups of users rather than single individuals, together with Catalano, Dent, Malone-Lee, Neven, and Smart, we introduced in [ABC+11, ACD+06] a new primitive called wildcarded identity-based encryption, or WIBE for short. It allows a sender to encrypt messages to a whole range of receivers whose identities match a certain pattern defined through a sequence of fixed strings and wildcards, where any string can take the place of a wildcard in a matching identity.

To better understand the concept of WIBE, consider the scenario where there is some kind of organizational hierarchy, such as a hierarchy of email addresses at an university of the form `user@dept.univ.edu`. Now suppose that we wish to send an encrypted email to all the users of the computer science department, which may include several personal addresses. If we use a standard HIBE, then one would need to encrypt the message to each user individually. On the other hand, in a WIBE scheme, we can easily achieve this goal by encrypting to the pattern `*@cs.univ.edu`.

3.4.1 Definition

As per [ACD+06] (whose full version is described in Appendix E), a WIBE scheme is a generalization of a HIBE in which users can decide at the time of encryption whether to make the ciphertext decryptable by a group of users whose identities match a certain pattern. More formally, a WIBE scheme is defined by a tuple of algorithms $\mathsf{WIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$. The Setup, KeyDer, and Dec algorithms are defined exactly as in a HIBE. On the other hand, the encryption algorithm Enc takes as input a pattern $P \in (\mathsf{ID} \cup *)^{\ell}$, the master public key mpk and the message m and returns a ciphertext for P. Such a pattern may contain a special wildcard symbol $*$ at some levels. An identity $id = (id_1, \ldots, id_\ell) \in \mathsf{ID}^{\ell}$ is said to *match* a pattern $P \in (\mathsf{ID} \cup *)^{\ell'}$, denoted as $id \in_* P$, if and only if $\ell \leq \ell'$ and $\forall i = 1, \ldots, \ell$: $id_i = P_i$ or $P_i = *$. Note that any ancestor of a matching identity is also a matching identity under this definition, which seems reasonable given that any ancestor can derive the secret key of a matching descendant identity.

For correctness, it is required that for all honestly generated master keys $(mpk, msk) \xleftarrow{R} \mathsf{Setup}$, for all messages $m \in \mathcal{M}$, all patterns $P \in (\mathsf{ID} \cup *)^{\ell'}$ and all identities $id \in \mathsf{ID}^{\ell}$ such that $id \in_* P$, $m \leftarrow \mathsf{Dec}(\mathsf{KeyDer}(msk, id), \mathsf{Enc}(mpk, P, m))$ holds with overwhelming probability. The security of WIBE schemes against adaptive-pattern chosen-plaintext attacks can be defined analogously to those of HIBE schemes in Section 3.2.2, except that the adversary outputs a challenge pattern instead of an identity at the end of the first phase. To exclude trivial attacks, the adversary is not allowed to query the key derivation oracle on any identity that matches the challenge pattern. Moreover, similar changes also apply to the case of selective-pattern chosen-plaintext attacks.

3.4.2 Constructions

In Appendix E, several instantiations of WIBE schemes are introduced based on existing HIBE constructions, such as the BB-HIBE scheme described in Figure 3.5 and the HIBE schemes due to Waters [Wat05] and to Boneh, Boyen, and Goh [BBG05]. As the WIBE scheme based on BB-HIBE (to which we refer as the BB-WIBE) illustrates well the idea behind these constructions, let us now recall it here.

The BB-WIBE *scheme.* To understand how the BB-HIBE scheme in Figure 3.6 can be transformed into a WIBE scheme, let us consider a particular case in which we want to encrypt to pattern $P = (id_1, id_2, *)$ where id_1 and id_2 are two arbitrary identities in the identity space ID. In the BB-HIBE scheme, the ciphertext for an identity (id_1, id_2, id_3) has the form $(C_1, C_{2,1}, C_{2,2}, C_{2,3}, C_3) = (g_1^r, (u_1 \cdot u_0^{id_1})^r, u_2 \cdot u_0^{id_2})^r, (u_3 \cdot u_0^{id_3})^r, m \cdot \hat{e}(h_1, g_2)^r)$. In order to allow decryption by any user having id_1 and id_2 as the first two components and an arbitrary value in the third component (including ε), the first step is to leave the computation of $C_{2,1}$ and $C_{2,2}$ unchanged and to partition $C_{2,3}$ into two parts $C_{2,3,0} = u_0^r$ and $C_{2,3,1} = u_3^r$, which are independent of the third identity component. This way, any user that matches the pattern can reconstitute the original BB-HIBE ciphertext by combining these two parts by plugging in its own third identity component. That is, it simply computes $C_{2,3}$ as $C_{2,3,0} \cdot C_{2,3,1}^{id_3}$. Unfortunately,

this modification is not enough to obtain a secure WIBE as the element $C_{2,3,0} = u_0^r$ can be used to change the recipient identity for non-wildcard positions as well. This is because the same value u_0 is used in the computation of each $C_{2,i}$. To overcome this problem and obtain a secure WIBE scheme, we further need to modify the original BB-HIBE scheme so that a different value $u_{0,i}$ is associated with each level. The resulting scheme, which we call BB-WIBE, is described in Figure 3.6.

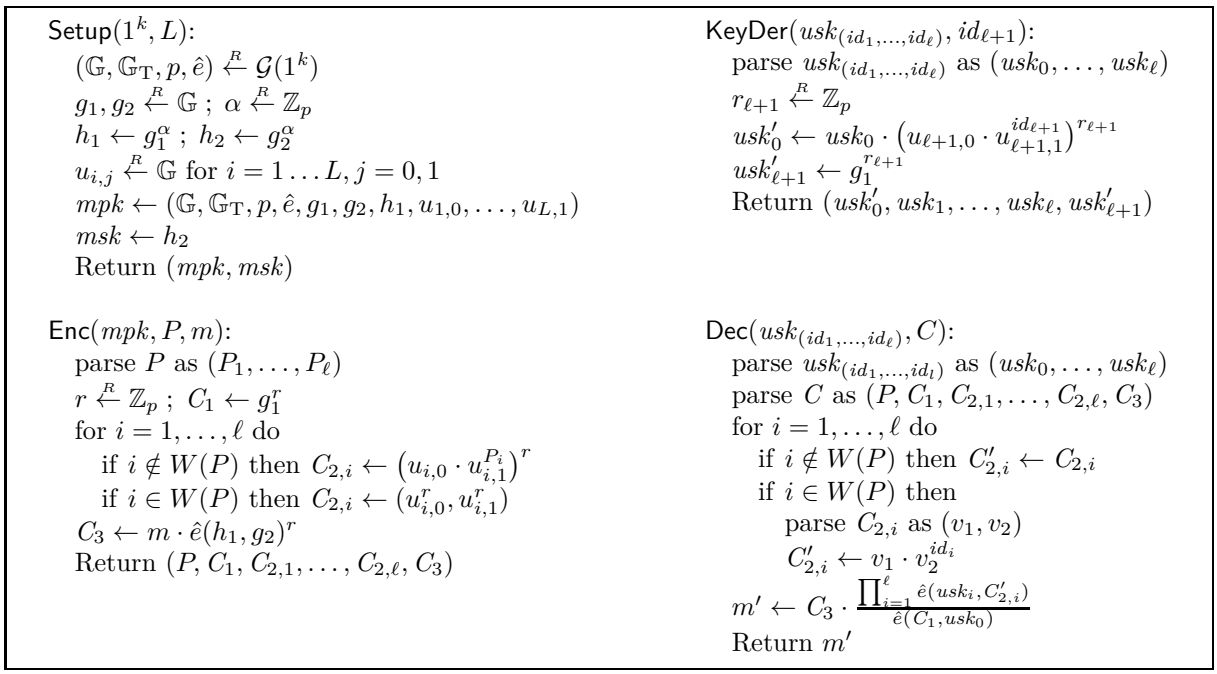


Figure 3.6: The BB-WIBE scheme in [ACD+06] and Appendix E, where $\mathcal{G}$ is a pairing parameter generator and $W(P) = \{1 \leq i \leq \ell \ : \ P_i = *\}$ denotes the set of wildcard positions in P. The description of KeyDer assumes $usk_\varepsilon = (usk_0) = (msk) = (h_2)$.

Security of BB-WIBE. Let $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ be the parameters generated by pairing parameter generator $\mathcal{G}$. If the BDDH problem described in Section 1.3 is hard with respect to $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$, then, as shown in Appendix E, the BB-WIBE scheme in Figure 3.6 is a secure against selective-pattern chosen-plaintext attacks.

Contributions. In addition to proposing the concept of wildcarded identity-based encryption and instantiations based on the HIBE schemes in [BB04a, BBG05, Wat05], we also propose in Appendix E a generic transformation in the random oracle model that converts any selective-pattern chosen-plaintext-secure WIBE scheme into a WIBE that is secure against adaptive-pattern chosen-plaintext attacks.

3.4.3 IBE with wildcard key derivation

Even though the concept of WIBE allows the sender to have a more fine-grained control over the list of recipients, its key delegation mechanism is similar to that of a hierarchical identity-based encryption scheme, which is not as flexible as one would like. In [AKN07, AKN08], Kiltz, Neven, and I introduce a related primitive called identity-based encryption with wildcard key derivation (WKD-IBE, or "wicked IBE") that enhances the concept of hierarchical identity-based encryption (HIBE) by allowing more general key delegation patterns. In a WKD-IBE, a secret

key is derived for a vector of identity strings, where entries can be left blank using a wildcard. This key can then be used to derive keys for any pattern that replaces wildcards with concrete identity strings. For example, one may want to allow the university's head system administrator to derive secret keys (and hence the ability to decrypt) for all departmental sysadmin email addresses `sysadmin@*.univ.edu`, where `*` is a wildcard that can be replaced with any string. In addition to providing appropriate security notions and provably secure instantiations for the new primitive, we also present a generic construction of identity-based broadcast encryption from any WKD-IBE scheme.

3.5 Public-key encryption with keyword search

One of the mains applications of IBE, WIBE, and WKD-IBE schemes is that of intelligent email routing, where emails usually consist of some header information, a body, and a list of keywords. In the scenario where a user may use different electronic devices to read his email, this user may prefer emails to be routed to his devices depending on the associated keywords. For example, a user may like to receive emails with the keyword "urgent" on his pager, emails with the keyword "agenda" on his PDA, and all other emails on his desktop computer. Though existing mail server software could be updated to provide this type of service for plain, unencrypted email, routing becomes much harder when emails are encrypted. One option would be for the user to leave the list of keywords unencrypted. However, since the keyword information may already be considered sensitive by the user, this option may not be very useful in practice.

Public-key encryption with keyword search. To address the above problem, Boneh, Di Crescenzo, Ostrovsky and Persiano proposed in [BDOP04] the notion of public-key encryption with keyword search (PEKS for short) where the user can give the mail server or gateway some piece of trapdoor information that allows it to test whether a given keyword is among those in the list, without revealing any other information about the email to the gateway. The user can then use a standard public-key encryption scheme to encrypt the body of the email, and a PEKS scheme to separately encrypt each of the keywords.

3.5.1 Definition

PEKS scheme [BDOP04] is a tuple $\mathsf{PEKS} = (\mathsf{KeyGen}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ of algorithms. Via $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}$, the key generation algorithm produces a pair of public and private keys. The encryption algorithm PEKS takes as input a keyword w and the public key pk and returns a ciphertext $C \xleftarrow{R} \mathsf{PEKS}(pk, w)$. Via $t_w \xleftarrow{R} \mathsf{Td}(sk, w)$, the trapdoor extraction algorithm computes a trapdoor t_w for keyword w. The deterministic test algorithm $\mathsf{Test}(t_w, C)$ returns 1 if C is an encryption of w and 0 otherwise.

In order to be useful, a PEKS scheme needs to satisfy three important conditions [ABC+05a]: *correctness*, *privacy*, and *consistency*. The correctness condition states that, for all honestly generated keys $(pk, sk) \xleftarrow{R} \mathsf{KeyGen}$ and for all keywords w in the keyword space, $\mathsf{Test}(\mathsf{Td}(sk, w), \mathsf{PEKS}(pk, w)) = 1$ holds with overwhelming probability. The privacy condition (IND-CPA) for PEKS schemes is similar to the standard indistinguishability notion for HIBE schemes in Section 3.2.2 and asks that no PTA adversary should be able to distinguish with non-negligible probability between the encryption of two challenge keywords of its choice, even if it is allowed to obtain trapdoors for any non-challenge keywords. Finally, the (computational) consistency condition requires that no PTA adversary should be able to find two different keywords (w_0, w_1) such that $\mathsf{Test}(\mathsf{Td}(sk, w_0), \mathsf{PEKS}(pk, w_1)) = 1$ with non-negligible probability. For a formal definition, please refer to Appendix F.

3.5.2 Constructions

In addition to proposing the concept of PEKS, Boneh *et al.* also showed how to build a PEKS scheme using the Boneh-Franklin IBE scheme described in Figure 3.4 as a starting point. Let BDOP-PEKS and BF-IBE refer to the new PEKS and the Boneh-Franklin IBE schemes. More precisely, keywords in the BDOP-PEKS scheme are mapped to identities in the BF-IBE scheme, with a trapdoor in the former simply corresponding to a decryption key in the latter. To encrypt a keyword w in the BDOP-PEKS, one simply uses BF-IBE to the encrypt the all-zero message for identity w. To test whether a ciphertext encrypts a given keyword w in the BDOP-PEKS scheme, one simply uses the BF-IBE decryption algorithm with the key associated with w and tests whether the result equals the all-zero message.

$\mathsf{KeyGen}(1^k)$:
$(\mathbb{G}, \mathbb{G}_T, p, \hat{e}) \xleftarrow{R} \mathcal{G}(1^k)$; $g \xleftarrow{R} \mathbb{G}^*$; $s \xleftarrow{R} \mathbb{Z}_p^*$
$pk \leftarrow (\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^s)$; $sk \leftarrow (pk, s)$
Return (pk, sk)

$\mathsf{PEKS}(pk, w)$:
parse pk as $(\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^s)$
$r \xleftarrow{R} \mathbb{Z}_p^*$; $T \leftarrow \hat{e}(H_1(w), g^s)^r$
$C \leftarrow (g^r, H_2(T))$
Return C

$\mathsf{Td}(sk, w)$:
parse sk as $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^s), s)$
$t_w \leftarrow (pk, H_1(w)^s)$
Return t_w

$\mathsf{Test}(t_w, C)$:
parse t_w as $((\mathbb{G}, \mathbb{G}_T, p, \hat{e}, g, g^s), X)$
parse C as (U, V) ; $T \leftarrow \hat{e}(X, U)$
if $V = H_2(T)$ then return 1
else return 0

Figure 3.7: The BDOP-PEKS scheme [BDOP04], where $\mathcal{G}$ is a pairing parameter generator and $H_1 : \{0,1\}^* \to \mathbb{G}^*$ and $H_2 : \mathbb{G}_T \to \{0,1\}^k$ are random oracles.

Security of BDOP-PEKS. Let $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$ be the parameters generated by pairing parameter generator $\mathcal{G}$. If the BDH problem is hard with respect to $(\mathbb{G}, \mathbb{G}_T, p, \hat{e})$, then, as shown in [BDOP04], the BDOP-PEKS scheme in Figure 3.7 is correct and private (IND-CPA) in the random oracle model. The computational consistency of their scheme is proven in [ABC+05a, ABC+08] also in the random oracle model (see Appendix F).

Contributions. In addition to proving the computational consistency of the BDOP-PEKS scheme in [ABC+05a, ABC+08], we formally define computational and statistical relaxations of the existing notion of perfect consistency and provide a new PEKS scheme that is statistically consistent. Moreover, we also suggest three extensions of the basic notions of PEKS and IBE, namely anonymous hierarchical identity-based encryption, public-key encryption with temporary keyword search, and identity-based encryption with keyword search. For more details, please refer to Appendix F.

3.5.3 Generic transforms

As the intuition behind the BDOP-PEKS scheme shows, the notions of PEKS and IBE are closely related to each other. In fact, Boneh *et al.* informally suggested a simple transformation from IBE to PEKS in [BDOP04]. The transform, to which we refer as bdop-ibe-2-peks, takes as input an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathsf{PEKS} = (\mathsf{KeyGen}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme (i.e., $\mathsf{KeyGen} = \mathsf{Setup}$). The trapdoor t_w associated to keyword w is the secret key that the IBE scheme would assign to the identity w (i.e., $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$). A keyword w is PEKS-encrypted by IBE-encrypting the message 0^k for the identity w (i.e., $\mathsf{PEKS}(pk, w) = \mathsf{Enc}(pk,$

$w, 0^k)$). Finally, testing is done by checking that the ciphertext decrypts to 0^k (i.e., $\mathsf{Test}(t_w, C)$ returns 1 iff $\mathsf{Dec}(t_w, C) = 0^k$).

While [BDOP04] stops short of stating and proving a formal result, it is not hard to verify that if the starting IBE scheme IBE is anonymous under adaptive-identity chosen-plaintext attacks (ANO-CPA, cf. Section 3.2.3), then PEKS = bdop-ibe-2-peks(IBE) is IND-CPA. Unfortunately, as we show in Appendix F, there are IBE schemes for which the PEKS scheme outputted by bdop-ibe-2-peks is not computationally consistent even if the IBE scheme is semantically secure (IND-CPA) and anonymous (ANO-CPA). Intuitively, this is due to the fact that the decryption algorithm may behave oddly when the secret key used to decrypt does not correspond to the identity used during encryption.

To address the problem of oddly-behaving Dec algorithms, we propose in Appendix F a randomized variant of the bdop-ibe-2-peks transform. The new transform, to which we refer as new-ibe-2-peks, is similar to bdop-ibe-2-peks except that instead of always using 0^k as the encrypted message, the PEKS-encryption algorithm chooses and encrypts a random message $m \in \{0,1\}^k$ and appends m in the clear to the ciphertext. In more detail, the new-ibe-2-peks transform takes input an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathsf{PEKS} = (\mathsf{KeyGen}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme. (I.e. $\mathsf{KeyGen} = \mathsf{Setup}$.) The trapdoor associated to keyword w is the secret key that the IBE scheme would assign to the identity w. (I.e. $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$.) PEKS-encryption of keyword w is done as follows: $\mathsf{PEKS}(pk, w)$ picks $m \xleftarrow{R} \{0,1\}^k$, lets $C \xleftarrow{R} \mathsf{Enc}(pk, w, m)$, and returns (C, m) as the ciphertext. Finally, $\mathsf{Test}(t_w, (C, m))$ returns 1 iff $\mathsf{Dec}(t_w, C) = m$.

The bdop-ibe-2-peks transform

$\mathsf{KeyGen}(1^k)$:
  $(pk, sk) \xleftarrow{R} \mathsf{Setup}(1^k)$
  Return (pk, sk)

$\mathsf{Td}(sk, w)$:
  $t_w \xleftarrow{R} \mathsf{KeyDer}(sk, w)$
  Return t_w

$\mathsf{PEKS}(pk, w)$:
  $m \leftarrow 0^k$
  $C \xleftarrow{R} \mathsf{Enc}(pk, w, m)$
  Return C

$\mathsf{Test}(t_w, C)$:
  $m' \leftarrow \mathsf{Dec}(t_w, C)$
  if $m' = 0^k$ then return 1 else return 0

The new-ibe-2-peks transform

$\mathsf{KeyGen}(1^k)$:
  $(pk, sk) \xleftarrow{R} \mathsf{Setup}(1^k)$
  Return (pk, sk)

$\mathsf{Td}(sk, w)$:
  $t_w \xleftarrow{R} \mathsf{KeyDer}(sk, w)$
  Return t_w

$\mathsf{PEKS}(pk, w)$:
  $m \xleftarrow{R} \{0,1\}^k$
  $C \xleftarrow{R} \mathsf{Enc}(pk, w, m)$
  Return (C, m)

$\mathsf{Test}(t_w, (C, m))$:
  $m' \leftarrow \mathsf{Dec}(t_w, C)$
  if $m' = m$ then return 1 else return 0

Figure 3.8: The bdop-ibe-2-peks and new-ibe-2-peks transforms in [ABC+05a, ABC+08]. Both transforms take as input an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and return a PEKS scheme $\mathsf{PEKS} = (\mathsf{KeyGen}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$.

Security of new-ibe-2-peks. Let IBE be the starting IBE scheme and let PEKS = new-ibe-2-peks(IBE) be the resulting PEKS scheme obtained via the new-ibe-2-peks transform in Figure 3.8. As we show in Appendix F, if IBE is semantically secure under chosen-identity attacks (IND-CPA, cf. Section 3.2.2), then PEKS is computationally consistent. Further, if IBE is anonymous under adaptive-identity chosen-plaintext attacks (ANO-CPA, cf. Section 3.2.3), then PEKS is also IND-CPA-secure.

Security of bdop-ibe-2-peks. As mentioned above, the PEKS scheme outputted by bdop-ibe-2-peks may not be computationally consistent in cases where the IBE decryption algorithm behaves oddly when the incorrect secret key is used during decryption. However, as we observe in Appendix G, the bdop-ibe-2-peks transform can provide consistency if the starting IBE scheme is robust. More precisely, let IBE denote the starting IBE scheme and let PEKS = bdop-ibe-2-peks(IBE) be the resulting PEKS scheme obtained via the new-ibe-2-peks transform in Figure 3.8. As we show in Appendix G, if IBE is weakly robust (WROB-CPA, cf. Section 3.2.4), then PEKS is computationally consistent. Moreover, if IBE is anonymous under adaptive-identity chosen-plaintext attacks (ANO-CPA, cf. Section 3.2.3), then PEKS is IND-CPA-secure.

3.6 Further extensions

As the transformation from IBE to PEKS above shows, IBE schemes can be quite useful for the construction of other seemingly unrelated primitives. In [ACF09], Catalano, Fiore, and I exploit yet another property of IBE schemes. More precisely, we propose a methodology to construct verifiable random functions from a class of identity based encryption schemes that we call VRF-suitable. Informally, an IBE is VRF-suitable if it provides what we call *unique key derivation* (by which the secret key associated with an identity id is uniquely defined when given the public key) and it satisfies an additional property that we call pseudorandom decapsulation. In a nutshell, pseudorandom decapsulation means that if one decrypts a ciphertext C, produced with respect to an identity id, using the decryption key corresponding to any other identity id' the resulting value looks random to a polynomially bounded observer. Interestingly, we show that most known IBE schemes already achieve pseudorandom decapsulation. Our construction is of interest both from a theoretical and a practical perspective. Indeed, apart from establishing a connection between two apparently unrelated primitives, our methodology is *direct* in the sense that, in contrast to most previous constructions, it avoids the inefficient Goldreich-Levin hardcore bit transformation.

Conclusion

In this thesis, we explored mechanisms for reducing the amount of trust in third parties based on passwords and identities. In the password-based setting, we presented several key exchange protocols which remain secure in spite of the low entropy of the secret keys used for authentication. In the identity-based setting, we considered new security requirements, such as anonymity and robustness, as well as new extensions allowing senders to have better control over the decryption capabilities of the receiver, such as WIBE and PEKS. While these results contribute significantly to the state of the art, there are still several interesting research directions to be explored in these areas.

In the password-based setting, one important open problem is to design *efficient* PAKE schemes achieving adaptive security, in which adversaries can corrupt users at any time and learn their internal states. Though the security model achieved by these schemes is more realistic, current adaptively secure PAKE constructions are either too inefficient or only proven secure in idealized models, such as the random oracle model, where certain building blocks such as hash functions are assumed to behave in ideal way. Therefore, it is important to overcome these limitations and design a practical PAKE scheme achieving adaptive security without having to resort to any idealized models.

In addition to adaptive security, another interesting research direction is this area is to design practical PAKE schemes in the group setting which can tolerate network failures and password mistypes and that can remain secure even in the presence of malicious insiders. Currently, the vast majority of the PAKE schemes do not take these cases into account.

In the identity-based setting, one interesting research direction is to reduce the gap between the existing models used in security proofs and the actual environment in which these cryptosystems are deployed. For instance, most of the existing security models assume that no information about the secret key or the randomness used to encrypt the message is leaked during the encryption process. However, it is well known that this is not always true in practice as the adversary may be able to learn partial information about these secrets using different types of side-channel attacks, such as power-consumption, fault, or time analyses. As a result, a cryptographic algorithm which is perfectly secure in theory can be completely insecure in practice if improperly implemented. In order to decrease this gap between theory and practice, it is thus important to design cryptographic schemes which are leakage-resilient in the sense that they remain secure even when the adversary is capable of learning partial information about the secret key or randomness used in these schemes. Although there have been several recent proposals of leakage-resilient identity-based cryptosystems schemes (e.g., [DHLAW10, LLW11, LRW11]) in the literature, most of these schemes either are not very inefficient or do not take into account all possible sources of side-channel attacks.

Another important direction in the identity-based setting is to design more advanced forms of encryption schemes, allowing for more complex decryption policies. In such encryption schemes, the capability of a receiver to decrypt a ciphertext may depend on the attributes associated with the decryption key and on the policy specified by the ciphertext. It is worth pointing out

that certain forms of advanced encryption schemes already exist in the literature, such as the functional encryption schemes of Lewko *et al.* [LOS+10], but they usually lack in efficiency or only support limited classes of functions and policies. Hence, designing more efficient functional encryption schemes which allow for more complex decryption policies still remains an important open problem.

Finally, in addition to the settings considered in this thesis, there are other interesting venues for reducing the amount of trust in third parties. Among these, perhaps the most promising direction is to use fully-homomorphic encryption schemes, such as the one by Brakerski, Gentry, and Vaikuntanathan in [BGV11], for performing arbitrary computations on encrypted data.

Part II

Personal publications

Books

1. M. Abdalla and P. Barreto, *LATINCRYPT 2010: 1st International Conference on Cryptology and Information Security in Latin America*, **LNCS 6212**, Springer, Puebla, Mexico, August 2010.

2. M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, *ACNS 2009: 7th International Conference on Applied Cryptography and Network Security*, **LNCS 5536**, Springer, Paris-Rocquencourt, France, Juin 2009.

Refereed journal papers

1. M. Abdalla, J. Birkett, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, J. Schuldt, and N. Smart, *Wildcarded Identity-Based Encryption*, **Journal of Cryptology**, 24(1):42–82, January 2011.

2. M. Abdalla, E. Kiltz, and G. Neven, *Generalized Key Delegation for Hierarchical Identity-Based Encryption*, **IET Information Security**, 2(3):67–78, September 2008.

3. M. Abdalla, J.H. An, M. Bellare, and C. Namprempre, *From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security*, **IEEE Transactions on Information Theory**, 54(8): 3631–3646, August 2008.

4. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, *Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions*, **Journal of Cryptology**, 21(3):350–391, July 2008.

5. T. Claveirole, M. Dias de Amorim, M. Abdalla, and Y. Viniotis, *Securing Wireless Sensor Networks Against Aggregator Compromises*, **IEEE Communications Magazine**, 46(4):134–141, April 2008.

6. C. Namprempre, G. Neven, and M. Abdalla, *A Study of Blind Message Authentication Codes*, **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, E90-A(1):75–82.

7. M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval, *Strong Password-Based Authentication in TLS using the Three-Party Group Diffie-Hellman Protocol*, **International Journal of Security and Networks**, Inderscience, 2(3/4):284–296, 2007.

8. M. Abdalla, P.-A. Fouque, and D. Pointcheval, *Password-Based Authenticated Key Exchange in the Three-Party Setting*, **IEE Proceedings**, 153(1):27–39, March 2006.

9. M. Abdalla, Y. Shavitt, and A. Wool., *Key management for restricted multicast using broadcast encryption.*, **IEEE/ACM Transactions on Networking**, 8(4):443–454, August 2000.

Refereed international conference papers

1. M. Abdalla, C. Chevalier, L. Granboulan, and D. Pointcheval, *Contributory Password-Authenticated Group Key Exchange with Join Capability*, **Topics in Cryptology - CT-RSA 2011**, LNCS 6558, pp. 142–160, Springer.

2. M. Abdalla, M. Bellare, and G. Neven, *Robust Encryption*, **Theory of Cryptography Conference – TCC 2010**, LNCS 5978, pp. 480–497, Springer.

3. M. Abdalla, C. Chevalier, M. Manulis, and D. Pointcheval, *Flexible Group Key Exchange with On-Demand Computation of Subgroup Keys*, **3rd African International Conference on Cryptology – AFRICACRYPT 2010**, LNCS 6055, pp. 351–368, Springer.

4. M. Abdalla, C. Chevalier, and D. Pointcheval, *Smooth Projective Hashing for Conditionally Extractable Commitments*, **Advances in Cryptology – CRYPTO 2009**, LNCS 5677, pp. 671–689, Springer.

5. M. Abdalla, D. Catalano, C. Chevalier, and D. Pointcheval, *Password-Authenticated Group Key Agreement with Adaptive Security and Contributiveness*, **2nd African International Conference on Cryptology – AFRICACRYPT 2009**, LNCS 5580, pp. 254–271, Springer.

6. M. Abdalla, D. Catalano, and D. Fiore, *Verifiable Random Functions from Identity based Key Encapsulation*, **Advances in Cryptology – EUROCRYPT 2009**, LNCS 5479 pp. 554–571, Springer.

7. M. Abdalla, X. Boyen, C. Chevalier, and D. Pointcheval, *Distributed Public-Key Cryptography from Weak Secrets*, **Public Key Cryptography - PKC 2009**, LNCS 5443, pp. 139–159, Springer.

8. M. Abdalla, M. Izabachène, and D. Pointcheval, *Anonymous and Transparent Gateway-based Password-Authenticated Key Exchange*, **7th International Conference on Cryptology and Network Security - CANS 2008**, LNCS 5339, pp. 133–148, Springer.

9. M. Abdalla, D. Catalano, C. Chevalier, and D. Pointcheval, *Efficient Two-Party Password-Based Key Exchange Protocols in the UC Framework*, **Topics in Cryptology - CT-RSA 2008**, LNCS 4964, pp. 335–351, Springer.

10. M. Abdalla, E. Kiltz, and G. Neven, *Generalized Key Delegation for Hierarchical Identity-Based Encryption*, **12th European Symposium On Research In Computer Security - ESORICS 2007**, LNCS 4734, pp. 139–154.

11. M. Abdalla, A. Dent, J. Malone-Lee, G. Neven, D. H. Phan, and N. Smart, *Identity-based Traitor Tracing*, **Public Key Cryptography - PKC 2007**, LNCS 4450, pp. 361–376.

12. M. Abdalla, J.-M. Bohli, M. I. González-Vasco, and R. Steinwandt, *(Password) Authenticated Key Establishment: From 2-Party To Group*, **Theory of Cryptography Conference – TCC 2007**, LNCS 4392, pp. 499–514.

13. M. Abdalla and D. Pointcheval, *A Scalable Password-based Group Key Exchange Protocol in the Standard Model*, **Advances in Cryptology – ASIACRYPT 2006**, LNCS 4284, pp. 332–347.

14. M. Abdalla, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, and N. Smart, *Identity-Based Encryption Gone Wild*, **Automata, Languages and Programming, 33rd International Colloquium – ICALP 2006**, LNCS 4052, pp. 300–311, Springer.

15. M. Abdalla, E. Bresson, O. Chevassut, and D. Pointcheval, *Password-based Group Key Exchange in a Constant Number of Rounds*, **Public Key Cryptography - PKC 2006**, LNCS 3958, pp. 427–442.

16. M. Abdalla, E. Bresson, O. Chevassut, B. Möller, and D. Pointcheval, *Provably Secure Password-Based Authentication in TLS*, **1st ACM Symposium on InformAtion, Computer and Communications Security – ASIACCS 2006**, ACM Press.

17. M. Abdalla, C. Namprempre, and G. Neven, *On the (Im)possibility of Blind Message Authentication Codes*, **Topics in Cryptology - CT-RSA 2006**, LNCS 3860, pp. 262–279, Springer.

18. M. Abdalla, O. Chevassut, P.-A. Fouque, and D. Pointcheval, *A Simple Threshold Authenticated Key Exchange from Short Secrets*, **Advances in Cryptology – ASIACRYPT 2005**, LNCS 3788, pp. 566–584.

19. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier and H. Shi, *Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions*, **Advances in Cryptology – CRYPTO 2005**, LNCS 3621, pp. 205–222.

20. M. Abdalla and D. Pointcheval, *Interactive Diffie-Hellman assumptions with applications to password-based authentication*, **Financial Cryptography – FC 2005**, LNCS 3570, pp. 341–356, Springer.

21. M. Abdalla and D. Pointcheval, *Simple password-based encrypted key exchange protocols*, **Topics in Cryptology – CT-RSA 2005**, LNCS 3376, pp. 191–208, Springer.

22. M. Abdalla, P.-A. Fouque, and D. Pointcheval, *Password-based authenticated key exchange in the three-party setting*, **Public Key Cryptography – PKC 2005**, LNCS 3386, pp. 65–84, Springer.

23. M. Abdalla, O. Chevassut, and D. Pointcheval, *One-time verifier-based encrypted key exchange*, **Public Key Cryptography – PKC 2005**, LNCS 3386, pp. 47–64, Springer.

24. M. Abdalla and B. Warinschi, *On the minimal assumptions of group signature scheme*, **International Conference on Information and Communication Security – ICICS 2004**, LNCS 3269, pp. 1–13, Springer.

25. M. Abdalla, J.H. An, M. Bellare, and C. Namprempre, *From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security*, **Advances in Cryptology – EUROCRYPT 2002**, LNCS 2332, pp. 418–433, Springer.

26. M. Abdalla, M. Bellare, and P. Rogaway, *The Oracle Diffie-Hellman assumptions and an analysis of DHIES*, **Topics in Cryptology – CT-RSA 2001**, LNCS 2020, pp. 143–158, Springer.

27. M. Abdalla, S. Miner, and C. Namprempre, *Forward Security in Threshold Signature Schemes*, **Topics in Cryptology – CT-RSA 2001**, LNCS 2020, pp. 441–456, Springer.

28. M. Abdalla and M. Bellare, *Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques*, **Advances in Cryptology – ASIACRYPT 2000**, LNCS 1976, pp. 546–559, Springer.

29. M. Abdalla and L. Reyzin, *A new forward-secure digital signature scheme*, **Advances in Cryptology – ASIACRYPT 2000**, LNCS 1976, pp. 116–129, Springer.

30. M. Abdalla, Y. Shavitt, and A. Wool, *Towards making broadcast encryption practical*, **Financial Cryptography 1999**, LNCS 1648, pp. 140-157, Springer.

Contribution to standards

1. M. Abdalla, M. Bellare, and P. Rogaway, *DHIES: an encryption scheme based on the Diffie-Hellman problem*, Contributions to **IEEE P1363a**, September 1998. Also appears in **ANSI X9.63** and **ISO 18033-2**.

Part III

Appendix: Articles

Password-Based Cryptography

Appendix A:

Simple Password-Based Encrypted Key Exchange Protocols, CTRSA 2005

Michel Abdalla and David Pointcheval

This article proposes two simple constructions of password-based authenticated key exchange protocols in the random-oracle model based on the computational Diffie-Hellman problem. Both constructions are among the most efficient password-based authenticated key exchange schemes based on the encrypted key exchange protocol of Bellovin and Merritt.

Appendix B:

Smooth Projective Hashing for Conditionally Extractable Commitments, CRYPTO 2009

Michel Abdalla, Céline Chevalier, and David Pointcheval

This article shows how to generalize the password-based authenticated key exchange construction of Gennaro and Lindell so that it achieves adaptive security in the universal composability framework with erasures. The new construction is reasonably efficient and does not rely on random oracles. It also avoids the use of zero-knowledge proofs by building more complex smooth projective hash functions that can be efficiently associated to extractable commitment schemes.

Appendix C:

Password-based Group Key Exchange in a Constant Number of Rounds, PKC 2006

Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and David Pointcheval

This article proposes the first provably-secure password-based group key exchange protocol with a constant number of rounds. The new protocol is based on the Burmester-Desmedt group key exchange and is very efficient, having low communication and computational costs per user. Its security is proven in the random-oracle and ideal-cipher models, under the Decisional Diffie-Hellman assumption.

Appendix D:

A Scalable Password-based Group Key Exchange Protocol in the Standard Model, ASIACRYPT 2006

Michel Abdalla and David Pointcheval

This article provides the first construction of a constant-round password-based group key exchange protocol whose security proof does not rely on any idealized model. The new construction is based on the Gennaro-Lindell password-based authenticated key exchange construction and on the Burmester-Desmedt group key exchange and only assumes the existence of a common reference string. Like the Gennaro-Lindell scheme, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman and quadratic residuosity.

Appendix A

Simple Password-Based Encrypted Key Exchange Protocols

CT-RSA 2005
[AP05] with D. Pointcheval

Abstract : *Password-based encrypted key exchange are protocols that are designed to provide pair of users communicating over an unreliable channel with a secure session key even when the secret key or password shared between two users is drawn from a small set of values. In this paper, we present two simple password-based encrypted key exchange protocols based on that of Bellovin and Merritt. While one protocol is more suitable to scenarios in which the password is shared across several servers, the other enjoys better security properties. Both protocols are as efficient, if not better, as any of the existing encrypted key exchange protocols in the literature, and yet they only require a single random oracle instance. The proof of security for both protocols is in the random oracle model and based on hardness of the computational Diffie-Hellman problem. However, some of the techniques that we use are quite different from the usual ones and make use of new variants of the Diffie-Hellman problem, which are of independent interest. We also provide concrete relations between the new variants and the standard Diffie-Hellman problem.*

A.1 Introduction

Background. Keys exchange protocols are cryptographic primitives used to provide a pair of users communicating over a public unreliable channel with a secure session key. In practice, one can find several flavors of key exchange protocols, each with its own benefits and drawbacks. An example of a popular one is the SIGMA protocol [Kra03] used as the basis for the signature-based modes of the Internet Key Exchange (IKE) protocol. The setting in which we are interested in this paper is the 2-party symmetric one, in which every pair of users share a secret key. In particular, we consider the scenario in which the secret key is a password.

Password-based key exchange. Password-based key exchange protocols assume a more realistic scenario in which secret keys are not uniformly distributed over a large space, but rather chosen from a small set of possible values (a four-digit pin, for example). They also seem more convenient since human-memorable passwords are simpler to use than, for example, having additional cryptographic devices capable of storing high-entropy secret keys. The vast majority

of protocols found in practice do not account, however, for such scenario and are often subject to so-called *dictionary* attacks. Dictionary attacks are attacks in which an adversary tries to break the security of a scheme by a brute-force method, in which it tries all possible combinations of secret keys in a given small set of values (i.e., the dictionary). Even though these attacks are not very effective in the case of high-entropy keys, they can be very damaging when the secret key is a password since the attacker has a non-negligible chance of winning.

To address this problem, several protocols have been designed to be secure even when the secret key is a password. The goal of these protocols is to restrict the adversary's success to on-line guessing attacks only. In these attacks, the adversary must be present and interact with the system in order to be able to verify whether its guess is correct. The security in these systems usually relies on a policy of invalidating or blocking the use of a password if a certain number of failed attempts has occurred.

Encrypted key exchange. The seminal work in the area of password-based key exchange is the encrypted key exchange (EKE) protocol of Bellovin and Merritt [BM92]. In their protocol, two users execute an encrypted version of the Diffie-Hellman key exchange protocol, in which each flow is encrypted using the password shared between these two users as the symmetric key. Due to the simplicity of their protocol, several other protocols were proposed in the literature based on it [BR00, BCP03, BCP04, KI02, Mac02], each with its own instantiation of the encryption function. Our protocol is also a variation of their EKE protocol.

Minimizing the use of random oracles. One of our main goals is to provide schemes that are simple and efficient, but relying as little as possible on random oracles. Ideally, one would want to completely eliminate the need of random oracles as done in the KOY protocol [KOY01]. However, such protocols tend to be less efficient than those based on the EKE protocol of Bellovin and Merritt [BM92].

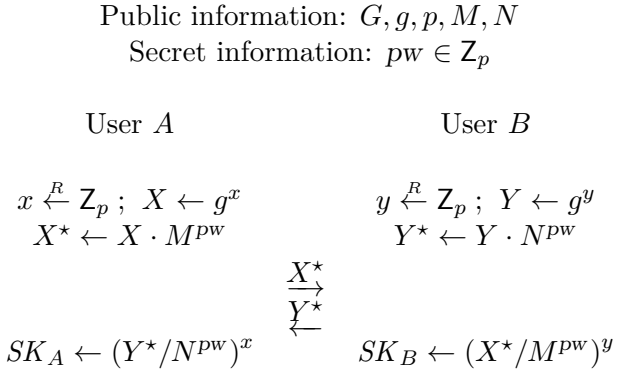


Figure A.1: An insecure password-based key exchange protocol.

To understand the difficulties involved in the design of protocols with few random oracles, let us consider the extreme case of the protocol in Figure A.1 in which no random oracles are used. Despite being secure against passive attacks, this protocol can be easily broken by an active adversary performing a man-in-the-middle attack. Such an adversary can easily create two different sessions whose session keys are related in a predictable manner. For instance, an adversary can do so by multiplying $X^\star$ by g^r for a known value r. The relation between the underlying session keys SK_A and SK_B is $SK_B = SK_A \cdot Y^r$. Hence, if the adversary learns the value of these two keys, it can perform an off-line dictionary attack using $Y = (SK_B / SK_A)^{-r}$ and $Y^\star$ to recover the password. Moreover, since the adversary can use arbitrary values for r, we cannot detect such attacks.

Protecting against related key attacks. In order to fix the problem in the protocol pre-

sented in Figure A.1 and prevent the adversary from altering the messages, one may be tempted to use message authentication code (MAC) algorithms for key derivation (e.g., by making the session key equal to $\mathsf{MAC}_{SK_A}(A, B, X^\star, Y^\star, 0)$) or key confirmation (e.g., by computing tags via $\mathsf{MAC}_{SK_A}(A, B, X^\star, Y^\star, 1)$). In fact, this is the approach used by Kobara and Imai in the construction of their password-authenticated key exchange protocol [KI02]. Unfortunately, this approach does not quite work.

Let us now explain the main problems with using MACs. First, the standard notion of security for MACs does not imply security against related key attacks. Hence, new and stronger security notions are required. Second, such new security notions may have to consider adversaries which are given access to a related-key tag-generation oracle. These are oracles that are capable of generating tags on messages under related keys, where the related key function is also passed as a parameter. This is actually the approach used in [KI02]. However, it is not clear whether such MACs can even be built. Such security notion, for instance, may completely rule out the possibility of using block-cipher-based MAC algorithms since similar security requirements in the context of block ciphers are known to be impossible to achieve [BK03]. Perhaps, hash-based MAC algorithms may be able to meet these goals, but that does not seem likely without resorting to random oracles, which would defeat the purpose of using MACs in the first place.

Simple constructions. In this paper, we deal with the problem of related-key attacks by using a single instance of a random oracle in the key derivation process. We present two simple constructions, whose only difference to one another is the presence of the password in the key derivation function. The presence of the password in the key derivation function is an important aspect, for example, when one wants to consider extensions to the distributed case, where each server only holds a share of password (see [DG03]).

Surprisingly, the techniques that we use to prove the security of these two constructions are quite different and so are the exact security results. While we are able to provide a tight security reduction for the scheme which includes the password in the key derivation phase, the same cannot be said about the other scheme, for which we can only prove its security in the non-concurrent scenario. However, the techniques that we use to prove the security of the latter are quite interesting and make use of new variants of the Diffie-Hellman problem.

New Diffie-Hellman assumptions. The new variants of the Diffie-Hellman problem that we introduce are called Chosen-Basis Diffie-Hellman assumptions due to the adversary's capability to choose some of the bases used in the definition of the problem. These assumptions are particular interesting when considered in the context of password-based protocols and we do expect to find applications for them other than the ones in this paper. Despite being apparently stronger than the standard Diffie-Hellman assumptions, we prove that this is *not* the case by providing concrete reductions to the computational Diffie-Hellman problem.

Contributions. In this paper, we address the issue of constructing efficient password-based encrypted key exchange protocols. Our main contributions are as follows.

Simple and efficient constructions in random oracle model. In this paper, we propose two new password-based encrypted key exchange protocols, called SPAKE1 and SPAKE2, both of which can be proven secure based on the hardness of the computational Diffie-Hellman problem in the random oracle model. Both protocols are comparable in efficiency to any of the existing EKE protocols, if not more efficient, and they only require one random oracle instance. This is contrast with existing EKE constructions, which require either a larger number of random oracle instances or additional ideal models, such as the ideal cipher model. Moreover, neither SPAKE1 nor SPAKE2 requires full domain hash functions or ideal ciphers onto a group, which are hard to implement efficiently. While one protocol is more suitable to scenarios in which the password is shared across several servers, the other enjoys better security properties.

New Diffie-Hellman assumptions. In proving the security of our protocols, we make use of new variations of the computational Diffie-Hellman assumption, called chosen-basis computational Diffie-Hellman assumptions. These new assumptions are of independent interest and we do expect to find new applications for it other than the ones in this paper. Reductions between the problems underlying the new assumptions and the standard computational Diffie-Hellman assumption are also provided.

Related work. Password-based authenticated key exchange has been extensively studied in the last few years [BPR00, BMP00, BCP03, BCP04, GL03, GL01, HK99, KOY01, MPS00, MSJ02, DG03] with the majority of them being submitted for inclusion in the IEEE P1363.2 standard [IEE04], a standard dealing with the issues of password-authenticated key agreement (e.g. EKE) and password-authenticated key retrieval. With the exception of [GL03, GL01, KOY01], all of these protocols are only proven secure in the random oracle model.

Perhaps, the related work that is closest to ours is the pretty-simple password-authenticated key exchange protocol of Kobara and Imai [KI02], whose proof of security is claimed to be in the "standard" model. Their protocol consists of EKE phase that is similar to the one used in our protocols followed by an authentication phase based on message authentication code (MAC) algorithms. However, the security model which they use is different from the standard one and hence their result only applies to their specific model. Moreover, as we pointed out above, their protocol needs a stronger security notion for the MAC algorithm and it is not clear whether such MACs can be built without resorting to random oracles, which would contradict their claims.

Organization. In Section A.2, we recall the security model for password-based authenticated key exchange. Next, in Section A.3, we present our new variants of the Diffie-Hellman problem and their relations to the computational Diffie-Hellman problem. Section A.4 then introduces the first of our password-based encrypted key exchange protocols, called SPAKE1, along with its proof of security. SPAKE1 is in fact based on one of the variants of the Diffie-Hellman problem introduced in Section A.3. Our second protocol, SPAKE2, is then presented in Section A.4 along with its security claims. In the appendix, we present proofs for several lemmas in Section A.3 as well as the proof of security for SPAKE2.

A.2 Security model for password-based key exchange

We now recall the security model for password-based authenticated key exchange of Bellare et al. [BPR00].

Protocol participants. Each participant in the password-based key exchange is either a client $C \in \mathcal{C}$ or a server $S \in \mathcal{S}$. The set of all users or participants $\mathcal{U}$ is the union $\mathcal{C} \cup \mathcal{S}$.

Long-lived keys. Each client $C \in \mathcal{C}$ holds a password pw_C. Each server $S \in \mathcal{S}$ holds a vector $pw_S = \langle pw_S[C] \rangle_{C \in \mathcal{C}}$ with an entry for each client, where $pw_S[C]$ is the transformed-password, as defined in [BPR00]. In this paper, we only consider the symmetric model, in which $pw_S[C] = pw_C$, but they may be different in general. pw_C and pw_S are also called the long-lived keys of client C and server S.

Protocol execution. The interaction between an adversary $\mathcal{A}$ and the protocol participants occurs only via oracle queries, which model the adversary capabilities in a real attack. During the execution, the adversary may create several instances of a participant. While in a concurrent model, several instances may be active at any given time, only one active user instance is allowed for a given intended partner and password in a non-concurrent model. Let U^i denote the instance i of a participant U and let b be a bit chosen uniformly at random. The query types available to the adversary are as follows:

- *Execute*(C^i, S^j): This query models passive attacks in which the attacker eavesdrops on honest executions between a client instance C^i and a server instance S^j. The output of this query consists of the messages that were exchanged during the honest execution of the protocol.

- *Send*(U^i, m): This query models an active attack, in which the adversary may tamper with the message being sent over the public channel. The output of this query is the message that the participant instance U^i would generate upon receipt of message m.

- *Reveal*(U^i): This query models the misuse of session keys by a user. If a session key is not defined for instance U^i or if a *Test* query was asked to either U^i or to its partner, then return $\perp$. Otherwise, return the session key held by the instance U^i.

- *Test*(U^i): This query tries to capture the adversary's ability to tell apart a real session key from a random one. If no session key for instance U^i is defined, then return the undefined symbol $\perp$. Otherwise, return the session key for instance U^i if $b = 1$ or a random key of the same size if $b = 0$.

NOTATION. Following [BR94a, BR95], an instance U^i is said to be *opened* if a query *Reveal*(U^i) has been made by the adversary. We say an instance U^i is *unopened* if it is not *opened*. We say an instance U^i has *accepted* if it goes into an accept mode after receiving the last expected protocol message.

PARTNERING. The definition of partnering uses the notion of session identifications (*sid*). More specifically, two instances U_1^i and U_2^j are said to be partners if the following conditions are met: (1) Both U_1^i and U_2^j accept; (2) Both U_1^i and U_2^j share the same session identifications; (3) The partner identification for U_1^i is U_2^j and vice-versa; and (4) No instance other than U_1^i and U_2^j accepts with a partner identification equal to U_1^i or U_2^j. In practice, the *sid* could be taken to be the partial transcript of the conversation between the client and the server instances before the acceptance.

FRESHNESS. The notion of freshness is defined to avoid cases in which adversary can trivially break the security of the scheme. The goal is to only allow the adversary to ask *Test* queries to *fresh* oracle instances. More specifically, we say an instance U^i is *fresh* if it has *accepted* and if both U^i and its partner are *unopened*.

SEMANTIC SECURITY. Consider an execution of the key exchange protocol P by an adversary $\mathcal{A}$, in which the latter is given access to the *Reveal*, *Execute*, *Send*, and *Test* oracles and asks a single *Test* query to a *fresh* instance, and outputs a guess bit b'. Such an adversary is said to win the experiment defining the semantic security if $b' = b$, where b is the hidden bit used by the *Test* oracle.

Let SUCC denote the event in which the adversary is successful. The ake**-advantage** of an adversary $\mathcal{A}$ in violating the semantic security of the protocol P and the **advantage function** of the protocol P, when passwords are drawn from a dictionary $\mathcal{D}$, are respectively

$$\mathbf{Adv}^{\mathrm{ake}}_{P,\mathcal{D}}(\mathcal{A}) = 2 \cdot \Pr[\,\mathrm{SUCC}\,] - 1 \quad \text{and} \quad \mathbf{Adv}^{\mathrm{ake}}_{P,\mathcal{D}}(t, R) = \max_{\mathcal{A}}\{\, \mathbf{Adv}^{\mathrm{ake}}_{P,\mathcal{D}}(\mathcal{A}) \,\}\ ,$$

where maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of queries to its oracles). The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size [ABR01].

A.3 Diffie-Hellman assumptions

In this section, we recall the definitions for the computational Diffie-Hellman assumption and introduce some new variants of it, which we use in the proof of security of simple password-based encrypted key exchange protocols. We also present some relations between these assumptions. In doing so, we borrow some of the notations in [ABR01].

A.3.1 Definitions

NOTATION. In the following, we assume a finite cyclic group G of prime order p generated by an element g. We also call the tuple $\mathbb{G} = (G, g, p)$ the represented group.

Computational Diffie-Hellman: CDH. The CDH assumption states that given g^u and g^v, where u and v were drawn at random from Z_p, it is hard to compute g^{uv}. Under the computational Diffie-Hellman assumption it might be possible for the adversary to compute something interesting about g^{uv} given g^u and g^v.

This can be defined more precisely by considering an Experiment $\mathbf{Exp}^{\mathrm{cdh}}_{\mathbb{G}}(\mathcal{A})$, in which we select two values u and v in Z_p, compute $U = g^u$, and $V = g^v$, and then give both U and V to an adversary $\mathcal{A}$. Let Z be the output of $\mathcal{A}$. Then, the Experiment $\mathbf{Exp}^{\mathrm{cdh}}_{\mathbb{G}}(\mathcal{A})$ outputs 1 if $Z = g^{uv}$ and 0 otherwise. Then, we define *advantage* of $\mathcal{A}$ in violating the CDH assumption as $\mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(\mathcal{A}) = \Pr[\,\mathbf{Exp}^{\mathrm{cdh}}_{\mathbb{G}}(\mathcal{A}) = 1\,]$ and the *advantage function* of the group, $\mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t)$, as the maximum value of $\mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(\mathcal{A})$ over all $\mathcal{A}$ with time-complexity at most t.

Chosen-basis computational Diffie-Hellman: CCDH. The chosen-basis computational Diffie-Hellman problem is a variation of the CDH problem. It considers an adversary that is given three random elements M, N and X in G and whose goal is to find a triple of values (Y, u, v) such that $u = \mathrm{CDH}(X, Y)$ and $v = \mathrm{CDH}(X/M, Y/N)$. The idea behind this assumption is that the adversary may be able to successfully compute either u (e.g., by choosing $Y = g$ and $u = X$) or v (e.g., by choosing $Y = g \cdot N$ and $v = X/M$), but not both. In fact, as we prove later, solving this problem is equivalent to solving the underlying computational Diffie-Hellman problem in $\mathbb{G}$. We now proceed with the formal definition.

Definition A.3.1 [CCDH] Let $\mathbb{G} = (G, g, p)$ be a represented group and let $\mathcal{A}$ be an adversary. Consider the following experiment, where M, N and X are elements in G,

Experiment $\mathbf{Exp}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A}, M, N, X)$
$(Y, u, v) \leftarrow \mathcal{A}(M, N, X)$
$u' \leftarrow \mathrm{CDH}(X, Y)$
$v' \leftarrow \mathrm{CDH}(X/M, Y/N)$
if $u = u'$ **and** $v = v'$ **then** $b \leftarrow 1$ **else** $b \leftarrow 0$
return b

Now define the *advantage* of $\mathcal{A}$ in violating the CCDH assumption with respect to (M, N, X), the *advantage* of $\mathcal{A}$, and the *advantage function* of the group, respectively, as follows:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A}, M, N, X) &= \Pr[\,\mathbf{Exp}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A}, M, N, X) = 1\,] \\
\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A}) &= \Pr_{M,N,X}\left[\,\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A}, M, N, X)\,\right] \\
\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(t) &= \max_{\mathcal{A}}\{\,\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(\mathcal{A})\,\},
\end{aligned}$$

where the maximum is over all $\mathcal{A}$ with time-complexity at most t. ♢

Password-based chosen-basis computational Diffie-Hellman: PCCDH. The password-based chosen-basis computational Diffie-Hellman problem is a variation of the chosen-basis computational Diffie-Hellman described above that is more appropriate to the password-based setting. The inputs to the problem and the adversarial goals are also somewhat different in this case so let us explain it.

Let $\mathcal{D} = \{1, \ldots, n\}$ be a dictionary containing n equally likely password values and let $\mathcal{P}$ be a public injective map $\mathcal{P}$ from $\{1, \ldots, n\}$ into Z_p. An example of an admissible map $\mathcal{P}$ is the one in which $\{1, \ldots, n\}$ is mapped into the subset $\{1, \ldots, n\}$ of Z_p. Now let us consider an adversary that runs in two stages. In the first stage, the adversary is given as input three random elements M, N and X in G as well as the public injective map $\mathcal{P}$ and it outputs a value Y in G. Next, we choose a random password $k \in \{1, \ldots, n\}$ and give it to the adversary. We also compute the mapping $r = \mathcal{P}(k)$ of the password k. The goal of the adversary in this second stage is to output a value K such that $K = \mathrm{CDH}(X/M^r, Y/N^r)$.

Note that an adversary that correctly guesses the password k in its first stage can easily solve this problem by computing $r = \mathcal{P}(k)$ and making, for instance, $Y = g \cdot N^r$ and $K = X/M^r$. Since we assume k to be chosen uniformly at random from the dictionary $\{1, \ldots, n\}$, an adversary that chooses to guess the password and follow this strategy can succeed with probability $1/n$.

The idea behind the password-based chosen-basis computational Diffie-Hellman assumption is that no adversary can do much better than the adversary described above. In fact, as we later prove, this should be the case as long as the computational Diffie-Hellman problem is hard in $\mathbb{G}$. We now proceed with the formal definition.

Definition A.3.2 [PCCDH] Let $\mathbb{G} = (G, g, p)$ be a represented group and let $\mathcal{A}$ be an adversary. Consider the following experiment, where M and N are elements in G, and $\mathcal{P}$ is a public injective map from $\{1, \ldots, n\}$ into Z_p,

$$
\begin{array}{l}
\textbf{Experiment } \mathbf{Exp}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, M, N, X', \mathcal{P}) \\
\quad (Y', st) \leftarrow \mathcal{A}(\mathsf{find}, M, N, X', \mathcal{P}) \\
\quad k \xleftarrow{R} \{1, \ldots, n\} \; ; \; r \leftarrow \mathcal{P}(k) \\
\quad (K) \leftarrow \mathcal{A}(\mathsf{guess}, st, k) \\
\quad X \leftarrow X'/M^r \; ; \; Y \leftarrow Y'/N^r \\
\quad \textbf{if } K = \mathrm{CDH}(X, Y) \textbf{ then } b \leftarrow 1 \textbf{ else } b \leftarrow 0 \\
\quad \textbf{return } b
\end{array}
$$

Now define the *advantage* of $\mathcal{A}$ in violating the PCCDH assumption with respect to $(M, N, X', \mathcal{P})$, the *advantage* of $\mathcal{A}$, and the *advantage function* of the group, respectively, as follows:

$$
\begin{aligned}
\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, M, N, X', \mathcal{P}) &= \Pr[\,\mathbf{Exp}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, M, N, X', \mathcal{P}) = 1\,] \\
\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, \mathcal{P}) &= \Pr_{M,N,X'}\left[\,\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, M, N, X', \mathcal{P})\,\right] \\
\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(t, \mathcal{P}) &= \max_{\mathcal{A}}\{\,\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, \mathcal{P})\,\},
\end{aligned}
$$

where the maximum is over all $\mathcal{A}$ with time-complexity at most t. $\diamondsuit$

Set password-based chosen-basis computational Diffie-Hellman: S-PCCDH. The set password-based chosen-basis computational Diffie-Hellman problem (S-PCCDH) is a multidimensional variation of the password-based chosen-basis computational Diffie-Hellman problem described above, in which the adversary is allowed to return not one key but a list of keys at the end of the second stage. In this case, the adversary is considered successful if the list of keys contains the correct value. We now proceed with the formal definition.

Definition A.3.3 [S-PCCDH] Let $\mathbb{G} = (G, g, p)$ be a represented group and let $\mathcal{A}$ be an adversary. Consider the following experiment, where M and N are elements in G, and $\mathcal{P}$ is a public injective map from $\{1, \ldots, n\}$ into Z_p,

Experiment $\mathbf{Exp}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, M, N, X', \mathcal{P})$
$(Y', st) \leftarrow \mathcal{A}(\mathsf{find}, M, N, X', \mathcal{P})$
$k \stackrel{R}{\leftarrow} \{1, \ldots, n\}$; $r \leftarrow \mathcal{P}(k)$
$(\mathcal{S}) \leftarrow \mathcal{A}(\mathsf{guess}, st, k)$
$X \leftarrow X'/M^r$; $Y \leftarrow Y'/N^r$
if $\text{CDH}(X, Y) \in \mathcal{S}$ **and** $|\mathcal{S}| \leq s$ **then** $b \leftarrow 1$ **else** $b \leftarrow 0$
return b

As above, we define the *advantage* of $\mathcal{A}$ in violating the S-PCCDH assumption with respect to $(M, N, X', \mathcal{P})$, the *advantage* of $\mathcal{A}$, and the *advantage function* of the group, respectively, as follows:

$$\begin{aligned}
\mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, M, N, X', \mathcal{P}) &= \Pr[\,\mathbf{Exp}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, M, N, X', \mathcal{P}) = 1\,] \\
\mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, \mathcal{P}) &= \Pr_{M,N,X'}\left[\,\mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, M, N, X', \mathcal{P})\,\right] \\
\mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},n,s}(t, \mathcal{P}) &= \max_{\mathcal{A}}\{\,\mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},n,s}(\mathcal{A}, \mathcal{P})\,\},
\end{aligned}$$

where the maximum is over all $\mathcal{A}$ with time-complexity at most t. ♢

A.3.2 Some relations

In this section, we first provide two relations between the above problems. The first result is meaningful for small n (polynomially bounded in the asymptotic framework). The second one considers larger dictionaries. Then, we show that these assumptions are implied by the classical computational Diffie-Hellman assumption. Finally, we also prove that the most general assumption is also implied by the classical computational Diffie-Hellman assumption.

Relations between the PCCDH and CCDH problems. The following two lemmas present relations between the PCCDH and CCDH problems. The first lemma, whose proof can be found in Appendix A.7, is oriented to the case of small dictionaries, for which n is polynomially-bounded. However, if n is large, super-polynomial in the asymptotic framework, or more concretely $n \geq 8/\epsilon$, then one should use the second lemma, whose proof can be easily derived from the proof of the first lemma (see Appendix A.7).

Lemma A.3.4 Let $\mathbb{G} = (G, g, p)$ be a represented group, let n be an integer, and let $\mathcal{P}$ be a public injective map from $\{1, \ldots, n\}$ into Z_p.

$$\frac{2}{n} \geq \mathbf{Adv}^{\text{pccdh}}_{\mathbb{G},n}(t, \mathcal{P}) \geq \frac{1}{n} + \epsilon \Longrightarrow \mathbf{Adv}^{\text{ccdh}}_{\mathbb{G}}(2t + 3\tau) \geq \frac{n}{128} \times \epsilon^3.$$

Lemma A.3.5 Let $\mathbb{G} = (G, g, p)$ be a represented group, let n be an integer, and let $\mathcal{P}$ be a public injective map from $\{1, \ldots, n\}$ into Z_p.

$$\mathbf{Adv}^{\text{pccdh}}_{\mathbb{G},n}(t, \mathcal{P}) \geq \epsilon \geq \frac{8}{n} \Longrightarrow \mathbf{Adv}^{\text{ccdh}}_{\mathbb{G}}(2t + 3\tau) \geq \frac{\epsilon^2}{32},$$

where τ denotes the time for an exponentiation in $\mathbb{G}$.

Relation between the CCDH and CDH problems. The following lemma, whose proof is in Appendix A.7, shows that the CCDH and CDH problems are indeed equivalent.

Lemma A.3.6 Let $\mathbb{G} = (G, g, p)$ be a represented group.

$$\mathbf{Adv}^{\mathrm{ccdh}}_{\mathbb{G}}(t) \leq \mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t + 2\tau),$$

where τ denotes the time for an exponentiation in $\mathbb{G}$.

Relation between the S-PCCDH and CDH problems. The following lemma, whose proof is in Appendix A.7, gives a precise relation between the S-PCCDH and CDH problems.

Lemma A.3.7 Let $\mathbb{G} = (G, g, p)$ be a represented group, let n and s be integers, and let $\mathcal{P}$ be a public injective map from $\{1, \ldots, n\}$ into Z_p.

$$\mathbf{Adv}^{\mathrm{s-pccdh}}_{\mathbb{G},n,s}(t, \mathcal{P}) \geq \frac{1}{n} + \epsilon \Longrightarrow \mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t') \geq \frac{n^2\epsilon^6}{2^{14}} - \frac{2s^4}{p},$$

where $t' = 4t + (18 + 2s)\tau$ and τ denotes the time for an exponentiation in $\mathbb{G}$. More concretely,

$$\mathbf{Adv}^{\mathrm{s-pccdh}}_{\mathbb{G},n,s}(t, \mathcal{P}) \geq \frac{1}{n} + \epsilon \geq \frac{1}{n} \times \left(1 + \frac{8(ns)^{2/3}}{p^{1/6}}\right) \Longrightarrow \mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t') \geq \frac{n^2\epsilon^6}{2^{15}}.$$

A.4 SPAKE1: a simple non-concurrent password-based encrypted key exchange

We now introduce our first protocol, SPAKE1, which is a *non-concurrent* password-based encrypted key exchange protocol, based on the multi-dimensional version of password-based chosen-basis computational Diffie-Hellman problem, S-PCCDH.

A.4.1 Description

SPAKE1 is a variation of the password-based encrypted key exchange protocol of Bellovin and Merritt [BM92], in which we replace the encryption function $\mathcal{E}_{pw}(.)$ with a simple one-time pad function. More specifically, whenever a user A wants to send the encryption of a value $X \in G$ to a user B, it does so by computing $X \cdot M^{pw}$, where M is an element in G associated with user A and the password pw is assumed to be in Z_p. The session identification is defined as the transcript of the conversation between A and B, and the session key is set to be the hash (random oracle) of the session identification, the user identities, and the Diffie-Hellman key. The password pw is *not* an input to the hash function. The full description of SPAKE1 is given in Figure A.2.

Correctness. The correctness of our protocol follows from the fact that, in an honest execution of the protocol, $K_A = K_B = g^{xy}$.

A.4.2 Security

As Theorem A.4.1 states, our non-concurrent password-based key exchange protocol is secure in the random oracle model as long as we believe that the S-PCCDH problem is hard in $\mathbb{G}$.

Public information: G, g, p, M, N, H
Secret information: $pw \in \mathsf{Z}_p$

User A		User B
$x \stackrel{R}{\leftarrow} \mathsf{Z}_p \; ; \; X \leftarrow g^x$		$y \stackrel{R}{\leftarrow} \mathsf{Z}_p \; ; \; Y \leftarrow g^y$
$X^\star \leftarrow X \cdot M^{pw}$		$Y^\star \leftarrow Y \cdot N^{pw}$
	$\stackrel{X^\star}{\longrightarrow}$	
	$\stackrel{Y^\star}{\longleftarrow}$	
$K_A \leftarrow (Y^\star / N^{pw})^x$		$K_B \leftarrow (X^\star / M^{pw})^y$
$SK_A \leftarrow H(A, B, X^\star, Y^\star, K_A)$		$SK_B \leftarrow H(A, B, X^\star, Y^\star, K_B)$

Figure A.2: SPAKE1: a simple non-concurrent password-based key exchange protocol.

Theorem A.4.1 Let $\mathbb{G}$ be a represent group and let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let SPAKE1 describe the password-based encrypted key exchange protocol associated with these primitives as defined in Figure A.2. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe},

$$\begin{aligned}
&\mathbf{Adv}^{\text{ake}}_{\mathsf{SPAKE},\mathcal{D}}(t, q_{\text{start}}, q^A_{\text{send}}, q^B_{\text{send}}, q_H, q_{\text{exe}}) \\
&\quad \leq \quad 2 \cdot (q^A_{\text{send}} + q^B_{\text{send}}) \cdot \mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},|\mathcal{D}|,q_H}(t', \mathcal{P}) + \\
&\qquad 2 \cdot \left(\frac{(q_{\text{exe}} + q_{\text{send}})^2}{2p} + q_H \; \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t + 2q_{\text{exe}}\tau + 3\tau) \right) ,
\end{aligned}$$

where q_H represents the number of queries to the H oracle; q_{exe} represents the number of queries to the *Execute* oracle; q_{start} and q^A_{send} represent the number of queries to the *Send* oracle with respect to the initiator A; q^B_{send} represents the number of queries to the *Send* oracle with respect to the responder B; $q_{\text{send}} = q^A_{\text{send}} + q^B_{\text{send}} + q_{\text{start}}$; $t' = t + O(q_{\text{start}}\tau)$; and τ is the time to compute one exponentiation in $\mathbb{G}$.

Since the S-PCCDH problem can be reduced to the CDH problem according to Lemma A.3.7, it follows that SPAKE1 is a secure non-concurrent password-based key exchange protocol in the random oracle model as long as the CDH problem is hard in $\mathbb{G}$, as stated in Corollary A.4.2.

Corollary A.4.2 Let $\mathbb{G}$ be a represent group and let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let SPAKE1 describe the password-based encrypted key exchange protocol associated with these primitives as defined in Figure A.2. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe},

$$\begin{aligned}
&\mathbf{Adv}^{\text{ake}}_{\mathsf{SPAKE},\mathcal{D}}(t, q_{\text{start}}, q^A_{\text{send}}, q^B_{\text{send}}, q_H, q_{\text{exe}}) \\
&\quad \leq \quad 2 \cdot \left(\frac{q^A_{\text{send}} + q^B_{\text{send}}}{|\mathcal{D}|} + \sqrt[6]{\frac{2^{14}}{|\mathcal{D}|^2} \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t') + \frac{2^{15} q_H^4}{|\mathcal{D}|^2 p}} \right) + \\
&\qquad 2 \cdot \left(\frac{(q_{\text{exe}} + q_{\text{send}})^2}{2p} + q_H \; \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t + 2q_{\text{exe}}\tau + 3\tau) \right) ,
\end{aligned}$$

where $t' = 4t + O((q_{\text{start}} + q_H)\tau)$ and the other parameters are defined as in Theorem A.4.1.

H oracle	– On hash query $H(q)$ (resp. $H'(q)$) for which there exists a record (q, r) in the list Λ_H (resp. Λ_H), return r. Otherwise, choose an element $r \in \{0,1\}^{l_k}$, add the record (q, r) to the list Λ_H (resp. Λ_H), and return r.

Figure A.3: Simulation of random oracles H and H'.

Send queries	– On a query $\mathit{Send}(A^i, \textbf{start})$, assuming A^i is in the correct state, we proceed as follows: **if** ActiveSessionIndex $\neq 0$ **then abort** $A^{\text{ActiveSessionIndex}}$ ActiveSessionIndex $= i$ $\theta \xleftarrow{R} \mathsf{Z}_p$; $\Theta \leftarrow g^\theta$; $\Theta^\star \leftarrow \Theta \cdot M^{pw}$ **return** $(A, \Theta^\star)$ – On a query $\mathit{Send}(B^i, (A, \Theta^\star))$, assuming B^i is in the correct state, we proceed as follows: $\phi \xleftarrow{R} \mathsf{Z}_p$; $\Phi \leftarrow g^\phi$; $\Phi^\star \leftarrow \Phi \cdot N^{pw}$ $K \leftarrow (\Theta^\star / M^{pw})^\phi$ $SK \leftarrow H(A, B, \Theta^\star, \Phi^\star, K)$ **return** $(B, \Phi^\star)$ – On a query $\mathit{Send}(A^i, (B, \Phi^\star))$, assuming A^i is in the correct state, we proceed as follows: $K \leftarrow (\Phi^\star / N^{pw})^\theta$ $SK \leftarrow H(A, B, \Theta^\star, \Phi^\star, K)$ ActiveSessionIndex $= 0$

Figure A.4: Simulation of *Send* oracle query.

Proof idea. Let $\mathcal{A}$ be an adversary against the semantic security of SPAKE. The idea is to use $\mathcal{A}$ to build adversaries for each of the underlying primitives in such a way that if $\mathcal{A}$ succeeds in breaking the semantic security of SPAKE, then at least one of these adversaries succeeds in breaking the security of an underlying primitive. Our proof consists of a sequence of hybrid experiments, starting with the real attack and ending in an experiment in which the adversary's advantage is 0, and for which we can bound the difference in the adversary's advantage between any two consecutive experiments.

Proof of Theorem A.4.1. Our proof uses a sequence of hybrid experiments, the first of which corresponds to the actual attack. For each experiment $\mathbf{Exp}_n$, we define an event Succ_n corresponding to the case in which the adversary correctly guesses the bit b involved in the *Test* query.

Experiment $\mathbf{Exp}_0$. This experiment corresponds to the real attack, which starts by choosing a random password pw. By definition, we have

$$\mathbf{Adv}^{\text{ake}}_{\mathsf{SPAKE}}(\mathcal{A}) \;=\; 2 \cdot \Pr[\,\text{Succ}_0\,] - 1 \tag{A.1}$$

Experiment $\mathbf{Exp}_1$. In this experiment, we simulate the *Execute*, *Reveal*, and *Send* oracles as in the real attack (see Figure A.4 and Figure A.5), after having chosen a random password pw. One can easily see that this experiment is perfectly indistinguishable from the real experiment. Hence,

$$\Pr[\,\text{Succ}_1\,] \;=\; \Pr[\,\text{Succ}_0\,] \tag{A.2}$$

Experiment $\mathbf{Exp}_2$. In this experiment, we simulate all oracles as in Experiment $\mathbf{Exp}_1$, except that we halt all executions in which a collision occurs in the transcript $((A, X^\star), (B, Y^\star))$. Since either $X^\star$ or $Y^\star$ was simulated and thus chosen uniformly at random, the probability of

Execute, *Reveal* and *Test* queries.

– On query $Reveal(U^i)$, proceed as follows:
if session key SK is defined for instance U^i
then return SK,
else return $\perp$.

– On query $Execute(A^i, B^j)$, proceed as follows:
$\theta \stackrel{R}{\leftarrow} \mathsf{Z}_p$; $\Theta \leftarrow g^\theta$; $\Theta^\star \leftarrow \Theta \cdot M^{pw}$
$\phi \stackrel{R}{\leftarrow} \mathsf{Z}_p$; $\Phi \leftarrow g^\phi$; $\Phi^\star \leftarrow \Phi \cdot N^{pw}$
$K \leftarrow \Theta^\phi$
$SK_A \leftarrow H(A, B, \Theta^\star, \Phi^\star, K)$; $SK_B \leftarrow SK_A$
return $((A, \Theta^\star), (B, \Phi^\star))$

– On query $Test(U^i)$, proceed as follows:
$SK \leftarrow Reveal(U^i)$
if $SK = \perp$ **then return** $\perp$
else
$b \stackrel{R}{\leftarrow} \{0,1\}$
if $b = 0$ **then** $SK' \leftarrow SK$ **else** $SK' \stackrel{R}{\leftarrow} \{0,1\}^{l_k}$
return SK'

Figure A.5: Simulation of *Execute*, *Reveal* and *Test* queries.

collisions in the transcripts is at most $(q_{\text{send}} + q_{\text{exe}})^2/(2p)$, according to the birthday paradox. Consequently,

$$\left|\Pr[\,\text{Succ}_2\,] - \Pr[\,\text{Succ}_1\,]\right| \;\leq\; \frac{(q_{\text{exe}} + q_{\text{send}})^2}{2p} \tag{A.3}$$

Experiment Exp$_3$. In this experiment, we replace the random oracle H by a secret one, for computing SK_A and SK_B for all sessions generated via an *Execute* oracle query. As the following lemma shows, the difference between the current experiment and the previous one is negligible as long as the CDH assumption holds. More precisely, we use a private random oracle H', and in the *Execute* oracle queries, one gets $SK_A, SK_B \leftarrow H'(A, B, \Theta^\star, \Phi^\star)$.

Lemma A.4.3 $\left|\Pr[\,\text{Succ}_3\,] - \Pr[\,\text{Succ}_2\,]\right| \leq q_H \cdot \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t + 2q_{\text{exe}}\tau + 3\tau)$.

Proof: The proof of Lemma A.4.3 uses the random self-reducibility of the Diffie-Hellman problem. Indeed, the only way for an execution to be altered by the above modification is if the adversary directly asks for $H(A, B, \Theta^\star, \Phi^\star, K)$, which will output something different from $H'(A, B, \Theta^\star, \Phi^\star)$, the answer of a *Reveal* query. But let us simulate the *Execute* oracle with a Diffie-Hellman instance (A, B), and thus $\Theta \leftarrow A \cdot g^\theta$ and $\Phi \leftarrow B \cdot g^\phi$. As a consequence, the above event means that $K = \text{CDH}(\Theta, \Phi) = \text{CDH}(A, B) \times A^\phi \times B^\theta \times g^{\phi\theta}$ is in the list of the queries asked to H: a random guess leads to $\text{CDH}(A, B)$. ▌

Experiment Exp$_4$. The goal of this experiment is to bound the advantage of the adversary during active attacks, in which the adversary has possibly generated the input of a *Send* oracle. To achieve this goal, we change the simulation of the *Send* oracle so that its output is chosen uniformly at random and independently of the password. The session key associated with each oracle is a bit string of appropriate length chosen uniformly at random and independently of input being provided to the *Send* oracle. The exact simulation of the *Send* oracle is as follows:

- On a query of type $(A^i, \textbf{start})$, we reply with $(A, X^\star = g^{x^\star})$ for a random $x^\star \in \mathsf{Z}_p$, if A^i is in the correct state. If another concurrent session already exists for user A, then we also terminate that session.
- On a query of type $(B^i, (A, X^\star))$, we reply with $(B, Y^\star = g^{y^\star})$ for a random $y^\star \in \mathsf{Z}_p$ and we set the session key SK_B to $H'(A, B, X^\star, Y^\star)$, if B^i is in the correct state.

- On a query of type $(A^i, (B, Y^\star))$, we set the session key SK_A to $H'(A, B, X^\star, Y^\star)$, if A^i is in the correct state.

As the following lemma shows, the adversary cannot do much better than simply guessing the password when distinguishing the current experiment from the previous one.

Lemma A.4.4 $\left|\Pr[\,\textsc{Succ}_4\,] - \Pr[\,\textsc{Succ}_3\,]\right| \leq (q^A_{\text{send}} + q^B_{\text{send}}) \cdot \mathbf{Adv}^{\text{s-pccdh}}_{\mathbb{G},|\mathcal{D}|,q_H}(t', \mathcal{P})$, where $t' = t + O(q_{\text{start}}\tau)$.

Proof: The proof of this lemma is based on a sequence of $q^A_{\text{send}} + q^B_{\text{send}} + 1$ hybrid experiments $\mathbf{Hybrid}_{3,j}$, where j is an index between 0 and $q_{AB} = q^A_{\text{send}} + q^B_{\text{send}}$. Let i be a counter for number of queries of the form $(B^k, (A, X^\star))$ or $(A^k, (B, Y^\star))$. That is, we do not count start queries (we do not increment this counter). We define Experiment $\mathbf{Hybrid}_{3,j}$ as follows:

- If $i \leq j$, then we processes the current *Send* query as in Experiment $\mathbf{Exp}_4$.
- If $i > j$, then we processes the current *Send* query as in Experiment $\mathbf{Exp}_3$.

It is clear from the above definition that experiments $\mathbf{Hybrid}_{3,0}$ and $\mathbf{Hybrid}_{3,q_{AB}}$ are equivalent to experiments $\mathbf{Exp}_3$ and $\mathbf{Exp}_4$, respectively. Now let P_j denote the probability of event Succ in Experiment $\mathbf{Hybrid}_{3,j}$. It follows that $\Pr[\,\textsc{Succ}_3\,] = P_0$ and $\Pr[\,\textsc{Succ}_4\,] = P_{q_{AB}}$. Moreover,

$$\left|\Pr[\,\textsc{Succ}_4\,] - \Pr[\,\textsc{Succ}_3\,]\right| \leq \sum_{j=1}^{q_{AB}} \left|P_j - P_{j-1}\right| .$$

The lemma will follow easily from bounding $\left|P_j - P_{j-1}\right|$. In order to so, consider the following algorithm D_j for the S-PCCDH problem in $\mathbb{G}$.

Algorithm D_j. Let $U = g^u$, $V = g^v$, and $W = g^w$ be random elements in G and let $\mathcal{P}$ be any injective map from $\{1, \ldots, n\}$ into Z_p. D_j starts running $\mathcal{A}$, simulating all its oracles. The *Reveal*, *Execute*, and *Test* oracles are simulated as in Experiment $\mathbf{Exp}_3$. The *Send* oracle is simulated as follows, Let i be the index of the current *Send* query.

- If the *Send* query is of the form $(A^k, \textbf{start}\ )$,
  - if $i \leq j$, then D_j replies with $(A, X^\star = W g^{x^\star})$ for a random $x^\star \in \mathsf{Z}_p$, if A^k is in the correct state. If another concurrent session already exists for user A, then D_j also terminates that session.
  - if $i > j$, then D_j processes it as in Experiment $\mathbf{Exp}_3$.
- If the query is of the form $(B^k, (A, X^\star))$,
  - if $i < j$, then D_j processes it as in Experiment $\mathbf{Exp}_4$.
  - if $i = j$, then D_j replies with $(B, Y^\star = W)$. It also returns $(st, Y' = X^\star)$ as the output of its find stage and waits for the input (st, k) of the guess stage. It then sets the password pw shared between A and B to $\mathcal{P}(k)$ and the session key SK_B to $H(A, B, X^\star, Y^\star, K_B)$, where $K_B = (X^\star / V^{pw})^{w-u\ pw}$. We note that st should contain all the necessary information for D_j to continue the execution of $\mathcal{A}$ and the simulation of its oracles in the guess stage. Let this be *Case B*.
  - if $i > j$, then D_j processes it as in Experiment $\mathbf{Exp}_3$.
- If the *Send* query is of the form $(A^k, (B, Y^\star))$,

- if $i < j$, then D_j processes it as in Experiment $\mathbf{Exp}_4$.
- if $i = j$, and A^k is in the correct state, then it returns $(st, Y' = Y^\star)$ as the output of its find stage and waits for the input (st, k) of the guess stage. Then, it sets the password pw shared between A and B to $\mathcal{P}(k)$ and the session key SK_A to $H(A, B, X^\star, Y^\star, K_A)$, where $K_A = (Y^\star/V^{pw})^{w+x^\star-u\ pw}$. Let this be *Case A*.
- if $i > j$, then D_j processes it as in Experiment $\mathbf{Exp}_3$.

Let K be the part of the input of H that is not present in H' and let $K_1, \ldots, K_{q_H}$ be the list of all such elements. When in Case A, D_j sets $K'_i = K_i/(Y'/V^{pw})^{x^\star}$ for $i = 1, \ldots, q_H$, where $x^\star$ is the value used to compute $X^\star$ in the crucial query. When in Case B, D_j simply sets $K'_i = K_i$. Finally, D_j outputs $K'_1, \ldots, K'_{q_H}$.

We note that in the above, the password is only defined at the j-th step and it is not used before that. Due to the non-concurrency, we do not need to know the password for simulating flows in Experiment $\mathbf{Exp}_4$. We only need it in Experiment $\mathbf{Exp}_3$.

Using the knowledge of u, v, and w in the above, it is clear that the processing of the *Send* queries matches that of Experiment $\mathbf{Hybrid}_{3,j-1}$. However, in the actual description of the S-PCCDH problem, we do not have access to these values. For this reason, the actual algorithm D_j replaces the random oracle H by a secret random oracle H' in the computation of SK_A and SK_B during the processing of the j-th *Send* query. More precisely, it computes SK_A and SK_B as $H'(A, B, X^\star, Y^\star)$. Moreover, we note that in this new scenario, the processing of the *Send* queries matches that of Experiment $\mathbf{Hybrid}_{3,j}$.

Probability analysis. Let AskH represent the event in which the adversary asks for $H(A, B, X^\star, Y^\star, K)$, where $K = \mathrm{CDH}(X^\star/U^{pw}, Y^\star/V^{pw})$ and either $X^\star$ or $Y^\star$ is involved in the crucial j-th query. We first observe that experiments $\mathbf{Hybrid}_{3,j-1}$ and $\mathbf{Hybrid}_{3,j}$ are identical if event AskH does not happen. Therefore, it follows that the probability difference $|P_j - P_{j-1}|$ is at most $\Pr[\,\mathrm{AskH}\,]$.

However, whenever event AskH happens, we know that the list of queries asked to H contains the key $K = \mathrm{CDH}(X^\star/U^{pw}, Y^\star/V^{pw})$ involved in the crucial query, and thus D_j will be able to successfully use $\mathcal{A}$ to help it solve the S-PCCDH problem. This is because K_A (Case A) or K_B (Case B) can be used to compute the solution $\mathrm{CDH}(W/U^{pw}, Y'/V^{pw})$ for the S-PCCDH problem as follows:

$$\begin{aligned} K_A &= \mathrm{CDH}(Y^\star/V^{pw}, Wg^{x^\star}/U^{pw}) = \mathrm{CDH}(W/U^{pw}, Y'/V^{pw}) \times (Y'/V^{pw})^{x^\star} \\ K_B &= \mathrm{CDH}(X^\star/V^{pw}, W/U^{pw}) = \mathrm{CDH}(W/U^{pw}, Y'/V^{pw}) \end{aligned}$$

Therefore, the list of candidates $K'_1, \ldots, K'_{q_H}$ outputted by D_j should contain the solution for the S-PCCDH problem whenever AskH happens. Hence, $\Pr[\,\mathrm{AskH}\,]$ is less than or equal to the success probability of D_j. The lemma follows easily from the fact that D_j has time-complexity at most t'. ❚❚

A.5 SPAKE2: a simple concurrent password-based encrypted key exchange

We now introduce our second protocol, SPAKE2, which is a *concurrent* password-based encrypted key exchange protocol, based on the computational Diffie-Hellman problem, CDH.

A.5.1 Description

SPAKE2 is a also variation of the password-based encrypted key exchange protocol of Bellovin and Merritt [BM92] and is almost exactly like SPAKE1. The only difference between the two is in the key derivation function, which also includes the password pw. More specifically, the session key in SPAKE2 is set to be the hash (random oracle) of the session identification, the user identities, the Diffie-Hellman key, and the password. In other words, $SK \leftarrow H(A, B, X^\star, Y^\star, pw, K)$. The session identification is still defined as the transcript of the conversation between A and B.

A.5.2 Security

As the following theorem states, our concurrent password-based key exchange protocol is secure in the random oracle model as long as the CDH problem is hard in G.

Theorem A.5.1 Let $\mathbb{G}$ be a represent group and let $\mathcal{D}$ be a uniformly distributed dictionary of size $|\mathcal{D}|$. Let SPAKE2 describe the password-based encrypted key exchange protocol associated with these primitives as defined in Section A.5.1. Then, for any numbers t, q_{start}, q^A_{send}, q^B_{send}, q_H, q_{exe},

$$\begin{aligned}
&\mathbf{Adv}^{\text{ake}}_{\mathsf{SPAKE2},\mathcal{D}}(t, q_{\text{start}}, q^A_{\text{send}}, q^B_{\text{send}}, q_H, q_{\text{exe}}) \\
&\quad \leq \quad 2 \cdot \left(\frac{q^A_{\text{send}} + q^B_{\text{send}}}{n} + \frac{(q_{\text{exe}} + q_{\text{send}})^2}{2p} \right) + \\
&\qquad\quad 2 \cdot \left(q_H \ \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t + 2q_{\text{exe}}\tau + 3\tau) + q_H^2 \ \mathbf{Adv}^{\text{cdh}}_{\mathbb{G}}(t + 3\tau) \right),
\end{aligned}$$

where the parameters are defined as in Theorem A.4.1.

Proof of Theorem A.5.1. The proof of security for SPAKE2 also uses a hybrid argument consisting of a sequence of experiments, the first of which corresponds to the actual attack. Since the proof of security for SPAKE2 is similar to that of SPAKE1, we only state the differences here.

In $\mathbf{Exp}_1$, we need to change the simulation of oracles in order to allow concurrent executions and to account for the presence of the password in the key derivation function. The claims remain unchanged.

In $\mathbf{Exp}_2$, no changes are required and thus the claims remain unchanged.

In $\mathbf{Exp}_3$, the only change comes from the fact the password is part of the input of H but not of H'. This change, however, does not affect the claims.

In $\mathbf{Exp}_4$, we also need to account for the password being part of the input of H but not of H'. In this case, however, the claims are different. In order to bound the difference in the adversary's success probability in experiments $\mathbf{Exp}_3$ and $\mathbf{Exp}_4$, we use a technique similar to the one used in the proof of security of the protocol MDHKE in [BCP04].

Let AskH_4 be the event in which the adversary directly asks the query $H(A, B, \Theta^\star, \Phi^\star, K)$ in experiment $\mathbf{Exp}_4$, where either $\Theta^\star$ or $\Phi^\star$ has been simulated during an active attack. Clearly, this is the only case in which the adversary can distinguish experiment $\mathbf{Exp}_4$ from experiment $\mathbf{Exp}_3$.

In order to upper-bound the probability of event AskH_4, let us first define Coll to be the event in which there exist two different values pw_1 and pw_2 such that the tuples $(X^\star, Y^\star, pw_i,$

$\mathrm{CDH}(X^\star/M^{pw_i}, Y^\star/N^{pw_i}))$ are present in the list of queries to H, for $i = 1, 2$. Then, by using a technique similar to that used in Lemma 5 in [BCP04], one can show that

$$\Pr[\,\textsc{Coll}\,] \leq q_H^2 \cdot \mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t + 3\tau) \ .$$

Now, let us consider the event $\textsc{AskH}_4$ given that the event Coll does not happen. That is, for each pair $(X^\star, Y^\star)$ involved in an active attack, there is at most one value of pw such that $(X^\star, Y^\star, pw, \mathrm{CDH}(X^\star/M^{pw}, Y^\star/N^{pw}))$ is in the list of queries to H. Since the password pw is not needed in the simulation of the oracles, we can postpone choosing its value until the very end of the simulation, at which moment we can detect whether the event $\textsc{AskH}_4$ has happened. Hence, the probability that the event $\textsc{AskH}_4$ happens given that the event Coll did not happen can be upper-bound by $\frac{q^A_{\mathrm{send}}+q^B_{\mathrm{send}}}{n}$. That is,

$$\Pr\left[\,\textsc{AskH}_4 \mid \overline{\textsc{Coll}}\,\right] \leq \frac{q^A_{\mathrm{send}} + q^B_{\mathrm{send}}}{n} \ .$$

As a result,

$$\begin{aligned}
\Pr[\,\textsc{AskH}_4\,] &\leq \Pr[\,\textsc{AskH}_4 \wedge \textsc{Coll}\,] + \Pr[\,\textsc{AskH}_4 \wedge \overline{\textsc{Coll}}\,] \\
&\leq \Pr[\,\textsc{Coll}\,] + \Pr\left[\,\textsc{AskH}_4 \mid \overline{\textsc{Coll}}\,\right] \\
&\leq q_H^2 \cdot \mathbf{Adv}^{\mathrm{cdh}}_{\mathbb{G}}(t + 3\tau) + \frac{q^A_{\mathrm{send}} + q^B_{\mathrm{send}}}{n} \ .
\end{aligned}$$

The proof of theorem follows immediately by noticing that the adversary's success probability is exactly 1/2 in this last experiment. ▮

Acknowledgments

The work described in this document has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability

A.6 Appendix: The splitting lemma

For simplicity, we reproduce here the splitting lemma presented in [PS00].

Lemma A.6.1 [Splitting Lemma] Let $A \subset X \times Y$ such that $\Pr[(x, y) \in A] \geq \epsilon$. For any $\alpha < \epsilon$, define

$$B = \left\{ (x, y) \in X \times Y \;\middle|\; \Pr_{y' \in Y}[(x, y') \in A] \geq \epsilon - \alpha \right\} \quad \text{and} \quad \bar{B} = (X \times Y) \backslash B,$$

then the following statements hold:

(i) $\Pr[B] \geq \alpha$

(ii) $\forall (x, y) \in B, \Pr_{y' \in Y}[(x, y') \in A] \geq \epsilon - \alpha$.

(iii) $\Pr[B \mid A] \geq \alpha/\epsilon$.

Proof: In order to prove statement (i), we argue by contradiction. Assume that $\Pr[B] < \alpha$. Then

$$\epsilon \leq \Pr[B] \cdot \Pr[A \mid B] + \Pr[\bar{B}] \cdot \Pr[A \mid \bar{B}] < \alpha \cdot 1 + 1 \cdot (\epsilon - \alpha) = \epsilon.$$

This implies a contradiction, hence the result. Statement (ii) is a straightforward consequence of the definition. We finally turn to the last assertion, using Bayes' law:

$$\Pr[B \mid A] = 1 - \Pr[\bar{B} \mid A] = 1 - \Pr[A \mid \bar{B}] \cdot \Pr[\bar{B}] / \Pr[A] \geq 1 - (\epsilon - \alpha)/\epsilon = \alpha/\epsilon.$$

∎

A.7 Appendix: Proof of lemmas

A.7.1 Proof of Lemma A.3.4

By definition of $\mathbf{Adv}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, \mathcal{P})$, we have

$$\Pr[\, b = 1 \,] \geq \frac{1}{n} + \epsilon,$$

where all the variables follow the distribution defined in **Experiment** $\mathbf{Exp}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A}, M, N, X', \mathcal{P})$, for uniformly distributed $M, N, X' \in G$ and $b \in \{0,1\}$. The probability space is thus defined by the variables ω, ρ, the random tapes of the adversary in the find-stage and the guess-stage respectively; M, N, X', the group elements; and k, the password:

$$\Omega_0 = \left\{ (\omega, \rho, M, N, X', k) \;\middle|\; \omega, \rho \xleftarrow{R} \{0,1\}^\star;\ (M, N, X') \xleftarrow{R} G^3;\ k \xleftarrow{R} \{1, \ldots, n\} \right\}.$$

By definition, we have

$$\Pr_{\Omega_0}[\, b = 1 \,] \geq \frac{1}{n} + \epsilon.$$

Applying the *splitting lemma* A.6.1, we get that if we split Ω_0 into $\Omega'_1 \times \Omega_1$:

$$\Omega'_1 = \left\{ \omega \;\middle|\; \omega \xleftarrow{R} \{0,1\}^\star \right\}$$

$$\Omega_1 = \left\{ (\rho, M, N, X', k) \;\middle|\; \rho \xleftarrow{R} \{0,1\}^\star\ ;\ (M, N, X') \xleftarrow{R} G^3\ ;\ k \xleftarrow{R} \{1, \ldots, n\} \right\},$$

there exists a set $\mathcal{S}_1 \subseteq \Omega'_1$ such that

$$\Pr_{\Omega'_1}[\, \mathcal{S}_1 \,] \geq \frac{\epsilon}{2}, \qquad \text{for any } \omega \in \mathcal{S}_1, \Pr_{\Omega_1}[\, b = 1 \,] \geq \frac{1}{n} + \frac{\epsilon}{2},$$

$$\Pr_{\Omega_0}[\, \omega \in \mathcal{S}_1 \mid b = 1 \,] \geq \frac{n\epsilon}{2 + 2n\epsilon}.$$

Applying again the *splitting lemma* A.6.1, we get that if we split Ω_1 into $\Omega'_2 \times \Omega_2$:

$$\Omega'_2 = \left\{ (M, N, X') \;\middle|\; (M, N, X') \xleftarrow{R} G^3 \right\}$$

$$\Omega_2 = \left\{ (\rho, k) \;\middle|\; \rho \xleftarrow{R} \{0,1\}^\star\ ;\ k \xleftarrow{R} \{1, \ldots, n\} \right\},$$

for each random tape $\omega \in \mathcal{S}_1$, there exists a set $\mathcal{S}_2(\omega) \subseteq \Omega'_2$ such that

$$\Pr_{\Omega'_2}[\, \mathcal{S}_2(\omega) \,] \geq \frac{\epsilon}{4}, \qquad \text{for any } (M, N, X') \in \mathcal{S}_2(\omega), \Pr_{\Omega_2}[\, b = 1 \,] \geq \frac{1}{n} + \frac{\epsilon}{4},$$

$$\Pr_{\Omega_1}\left[\,(M,N,X')\in\mathcal{S}_2(\omega)\mid b=1\,\right]\geq\frac{n\epsilon}{4+2n\epsilon}.$$

Let (U,V,X) be a random instance of the CCDH problem. We choose a random tape $\omega\stackrel{R}{\leftarrow}\{0,1\}^\star$, and two random indices $i<j\in\{1,\ldots,n\}$. Then, we define $r=\mathcal{P}(i)$ and $r'=\mathcal{P}(j)$, and then set $\delta=r'-r$, $M=U^{1/\delta}$, $N=V^{1/\delta}$, as well as $X'=X\times M^r$. Since (U,V,X) is a random triple, (M,N,X') is also uniformly distributed (and thus independently of r and r', and thus of i and j): with probability greater than $1/n+\epsilon$, $\mathcal{A}$ wins the **Experiment** $\mathbf{Exp}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A},M,N,X',\mathcal{P})$, with $k=i$, and the random tape (ω,ρ). In the favorable case, we have $\omega\in\mathcal{S}_1$ and $(M,N,X')\in\mathcal{S}_2(\omega)$ with probability

$$\frac{n\epsilon}{2+2n\epsilon}\times\frac{n\epsilon}{4+2n\epsilon}\geq\frac{1}{8}\times\left(\frac{n\epsilon}{1+n\epsilon}\right)^2.$$

Thereafter, with probability greater than $\epsilon/4$, $\mathcal{A}$ wins the **Experiment** $\mathbf{Exp}^{\mathrm{pccdh}}_{\mathbb{G},n}(\mathcal{A},M,N,X',\mathcal{P})$, with $k=j$, and the random tape (ω,ρ'):

$$\Pr[\,K=\mathrm{CDH}(X'/M^r,Y'/N^r)\wedge K'=\mathrm{CDH}(X'/M^{r'},Y'/N^{r'})\,]\geq\left(\frac{1}{n}+\epsilon\right)\times\frac{1}{8}\times\left(\frac{n\epsilon}{1+n\epsilon}\right)^2\times\frac{\epsilon}{4}.$$

Since we assumed, in the theorem, that $2/n\geq 1/n+\epsilon$, then $1+n\epsilon\leq 2$. Furthermore,

$$\begin{aligned}K &= \mathrm{CDH}(X'/M^r,Y'/N^r)=\mathrm{CDH}(X,Y)\\ K' &= \mathrm{CDH}(X'/M^{r'},Y'/N^{r'})=\mathrm{CDH}(X/M^\delta,Y/N^\delta)=\mathrm{CDH}(X/U,Y/V)\end{aligned}$$

This concludes the proof of this lemma. ▮

A.7.2 Proof of Lemma A.3.5

The proof of this lemma is similar to that of Lemma A.3.4 and is hence omitted in this version of the paper.

A.7.3 Proof of Lemma A.3.6

Let us consider an instance A, B for the Diffie-Hellman problem. We choose a random $b\in\mathsf{Z}_p$. Then we set $M\leftarrow A$, $N\leftarrow B$ and $X\leftarrow A^b$. We run an adversary $\mathcal{A}$ on this triple (M,N,X), and get (Y,u,v), which in case of success $u=\mathrm{CDH}(X,Y)$ and $v=\mathrm{CDH}(X/M,Y/N)$:

$$\begin{aligned}u &= \mathrm{CDH}(X,Y)=\mathrm{CDH}(A^b,Y)=\mathrm{CDH}(A,Y)^b\\ v &= \mathrm{CDH}(X/M,Y/N)=\mathrm{CDH}(A^{b-1},Y/B)=\mathrm{CDH}(A,Y/B)^{b-1}\\ &= \mathrm{CDH}(A,Y)^{b-1}/\mathrm{CDH}(A,B)^{b-1}\\ u^{b-1} &= \mathrm{CDH}(A,Y)^{b(b-1)}\\ v^b &= \mathrm{CDH}(A,Y)^{b(b-1)}/\mathrm{CDH}(A,B)^{b(b-1)}=u^{b-1}/\mathrm{CDH}(A,B)^{b(b-1)}\end{aligned}$$

Thus,

$$\mathrm{CDH}(A,B)^{b(b-1)}=u^{b-1}/v^b\quad\text{ and }\quad\mathrm{CDH}(A,B)=u^{1/b}v^{-1/(b-1)}.$$▮

A.7.4 Proof of Lemma A.3.7

In this proof, we use a general technique presented in [Sho97] for computing the solution of a given instance of a problem by finding a collision between the list of candidates for two related instances.

On a given A, B instance of the Diffie-Hellman problem, we use exactly the same argument as in the proof of Lemma A.3.4, running the S-PCCDH algorithm to break two instances of the CCDH problem: $U_1 = Ag^{u_1}$, $V_1 = Bg^{v_1}$ and $X_1 = A^{b_1}$, and $U_2 = Ag^{u_2}$, $V_2 = Bg^{v_2}$ and $X_2 = A^{b_2}$, for random values $u_1, u_2, v_1, v_2, b_1, b_2 \in \mathsf{Z}_p$. We then get four sets, $\mathcal{S}_1$, $\mathcal{S}'_1$, $\mathcal{S}_2$ and $\mathcal{S}'_2$, (instead of values) which contain candidates for $\mathrm{CDH}(X_1, Y_1)$, $\mathrm{CDH}(X_1/U_1, Y_1/V_1)$, $\mathrm{CDH}(X_2, Y_2)$ and $\mathrm{CDH}(X_2/U_2, Y_2/V_2)$ respectively. Therefore, with probability greater than $(n\epsilon^3/128)^2$, each set contains the correct value. Note that

$$\begin{aligned}
K_i &= \mathrm{CDH}(X_i, Y_i) = \mathrm{CDH}(A, Y_i)^{b_i} \\
K'_i &= \mathrm{CDH}(X_i/U_i, Y_i/V_i) \\
&= \mathrm{CDH}(X_i, Y_i) \times \mathrm{CDH}(U_i, V_i)/\mathrm{CDH}(X_i, V_i) \times \mathrm{CDH}(U_i, Y_i) \\
&= \mathrm{CDH}(X_i, Y_i) \times \mathrm{CDH}(Ag^{u_i}, Bg^{v_i})/\mathrm{CDH}(A^{b_i}, Bg^{v_i}) \times \mathrm{CDH}(Ag^{u_i}, Y_i) \\
&= \mathrm{CDH}(X_i, Y_i) \times \mathrm{CDH}(A, B)A^{v_i}B^{u_i}g^{u_i v_i}/\mathrm{CDH}(A, B)^{b_i}A^{v_i b_i} \times \mathrm{CDH}(A, Y_i)Y_i^{u_i} \\
&= \mathrm{CDH}(A, Y_i)^{b_i - 1} \times \mathrm{CDH}(A, B)^{1-b_i} \times A^{v_i(1-b_i)}(B/Y_i)^{u_i}g^{u_i v_i}
\end{aligned}$$

$$\begin{aligned}
\mathrm{CDH}(A, B)^{b_i - 1} &= \mathrm{CDH}(A, Y_i)^{b_i - 1} \times A^{v_i(1-b_i)}(B/Y_i)^{u_i}g^{u_i v_i}/K'_i \\
\mathrm{CDH}(A, B) &= \mathrm{CDH}(A, Y_i) \times A^{-v_i}(B/Y_i)^{u_i/(b_i-1)}g^{u_i v_i/(b_i-1)}/{K'_i}^{1/(b_i-1)} \\
&= \left({K_i}^{1/b_i}/{K'_i}^{1/(b_i-1)}\right) \times A^{-v_i}(B/Y_i)^{w_i}g^{w_i v_i},
\end{aligned}$$

where $w_i = u_i/(b_i - 1) \bmod p$.

With all the pairs in $\mathcal{S}_1 \times \mathcal{S}'_1$, we build the s^2 candidates for $\mathrm{CDH}(A, B)$, and we do the same with all the pairs in $\mathcal{S}_2 \times \mathcal{S}'_2$. The first collision in the two lists is outputted as the value for $\mathrm{CDH}(A, B)$. But for each pair $(k_i, k'_i) \in \mathcal{S}_1 \times \mathcal{S}'_1$, the candidate is

$$\alpha = \left({k_i}^{1/b_i}/{k'_i}^{1/(b_i-1)}\right) \times A^{-v_i}(B/Y_i)^{w_i}g^{w_i v_i} = \left(\frac{k_i}{K_i}\right)^{1/b_i} \times \left(\frac{K'_i}{k'_i}\right)^{1/(b_i-1)}.$$

If $k_i \neq K_i$, or $k'_i \neq K'_i$, it is totally random in G, because of the randomness of b_i. The probability of a wrong collision is thus bounded by $2s^4/p$. ∎

Appendix B

Smooth Projective Hashing for Conditionally Extractable Commitments

CRYPTO 2009
[ACP09] with C. Chevalier and D. Pointcheval

Abstract : *The notion of smooth projective hash functions was proposed by Cramer and Shoup and can be seen as special type of zero-knowledge proof system for a language. Though originally used as a means to build efficient chosen-ciphertext secure public-key encryption schemes, some variations of the Cramer-Shoup smooth projective hash functions also found applications in several other contexts, such as password-based authenticated key exchange and oblivious transfer. In this paper, we first address the problem of building smooth projective hash functions for more complex languages. More precisely, we show how to build such functions for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which smooth projective hash functions are known to exist. Next, we illustrate how the use of smooth projective hash functions with more complex languages can be efficiently associated to extractable commitment schemes and avoid the need for zero-knowledge proofs. Finally, we explain how to apply these results to provide more efficient solutions to two well-known cryptographic problems: a public-key certification which guarantees the knowledge of the private key by the user without random oracles or zero-knowledge proofs and adaptive security for password-based authenticated key exchange protocols in the universal composability framework with erasures.*

B.1 Introduction

In [CS02], Cramer and Shoup introduced a new primitive called smooth projective hashing and showed how to use it to generalize their chosen-ciphertext secure public-key encryption scheme [CS98]. The new abstraction not only provided a more intuitive description of the original encryption scheme, but also resulted in several new instantiations based on different security assumptions such as quadratic residuosity and N-residuosity [Pai99].

The notion of smooth projective hash functions (SPHF, [CS02], after slight modifications [GL03]) has been proven quite useful and has found applications in several other contexts, such as password-based authenticated key exchange (PAKE, [GL03]) and oblivious transfer [Kal05].

In the context of PAKE protocols, the work of Gennaro and Lindell abstracted and generalized (under various indistinguishability assumptions) the earlier protocol by Katz, Ostrovsky, and Yung [KOY01] and has become the basis of several other schemes [BCL$^+$05, AP06, BGS06]. In the context of oblivious transfer, the work of Kalai [Kal05] also generalized earlier protocols by Naor and Pinkas [NP01] and by Aiello, Ishai, and Reingold [AIR01].

To better understand the power of SPHF, let us briefly recall what they are. First, the definition of SPHF requires the existence of a domain X and an underlying NP language L such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. For instance, in the particular case of the PAKE scheme in [CHK$^+$05], the language L is defined as the set of triples $\{(c, \ell, m)\}$ such that c is an encryption of m with label ℓ under a public key given in the common reference string (CRS). The semantic security of the encryption scheme guarantees computational indistinguishability between elements from L and elements from X.

One of the key properties that make SPHF so useful is that, for a point $x \in L$, the hash value can be computed using either a *secret* hashing key hk, or a *public* projected key hp (depending on x [GL03] or not [CS02]) together with a witness w to the fact that $x \in L$. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$ and statistically indistinguishable from random for points $x \in X \setminus L$. Moreover, without the knowledge of the witness w to the fact that $x \in L$, the output of these functions on x is also pseudo-random.

The first main contribution of this paper is to extend the line of work on SPHF, the element-based version proposed by Gennaro and Lindell [GL03], to take into account more complex NP languages. More precisely, we show how to build SPHF for languages that can be described in terms of disjunctions and conjunctions of simpler languages for which SPHF are known to exist. For instance, let H_m represent a family of SPHF for the language $\{(c)\}$, where c is the encryption of m under a given public key. Using our tools, one can build a family of SPHF for the language $\{(c)\}$, where c is the encryption of either 0 or 1, by combining H_0 and H_1.

One of the advantages of building SPHF for more complex languages is that it allows us to simplify the design of the primitives to which they are associated. To demonstrate this, we consider in this paper the specific case of extractable commitment schemes. In most protocols in which extractable commitments are used, the capability of extracting the committed message usually depends on the commitment being properly generated. To achieve this goal and enforce the correct generation of the commitment, it is often the case that additional mechanisms, such as zero-knowledge proofs, may have to be used. This is the case, for instance, of several protocols where a specific public-key registration phase is required, such as most of the cryptographic protocols with dynamic groups (multisignatures [Bol03, LOS$^+$06], group signatures [DP06], etc). Such a framework is sometimes named *registered public-key model*, where a proof of knowledge of the secret key is required before any certification.

To be able to build more efficient extractable commitment schemes and avoid the use of possibly expensive concurrent zero-knowledge proofs, a second main contribution of this paper is to generalize the concept of extractable commitments so that extraction may fail if the commitment is not properly generated. More specifically, we introduce a new notion of L-extractable commitments in which extraction is only guaranteed if the committed value belongs to the language L and may fail otherwise. The main intuition behind this generalization is that, when used together with a SPHF for the language L, the cases in which extraction may fail will not be very important as the output of the SPHF will be statistically indistinguishable from random in such cases.

B.1.1 Applications

Registered Public-Key Setting. For many cryptographic protocols, for proving the security even when users can dynamically join the system, the simulator described in the security proof often needs to know the private keys of the authorized users, which is called the *registered public-key setting*, in order to avoid rogue-attacks [Bol03]. This should anyway be the correct way to proceed for a certification authority: it certifies a public key to a user if and only if the latter provides a proof of knowledge of the associated private key. However, in order to allow concurrency, intricate zero-knowledge proofs are required, which makes the certification process either secure in the random oracle model [BR93] only, or inefficient in the standard model.

In this paper, we show how SPHF with conditionally extractable commitments can help to solve this problem efficiently, in the standard model, by establishing a secure channel between the players, with keys that are either the same for the two parties if the commitment has been correctly built, or perfectly independent in the other case.

Adaptively-secure PAKE schemes. We thereafter study more involved key exchange schemes. In 1992, Bellovin and Merritt [BM92] suggested a method to authenticate a key exchange based on simple passwords, possibly drawn from a space so small that an adversary might enumerate off-line all possible values. Because of the practical interest of such a primitive, many schemes have been proposed and studied. In 2005, Canetti *et al.* [CHK$^+$05] proposed an ideal functionality for PAKE protocols, in the universal composability (UC) framework [Can01, CK02], and showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol. Though quite efficient, their protocol is not known to be secure against adaptive adversaries, where they can corrupt players at any time, and learn their internal states. The first ones to propose an adaptively-secure PAKE in the UC framework were Barak *et al.* [BCL$^+$05] using general techniques from multi-party computation (MPC). Though conceptually simple, their solution yields quite inefficient schemes.

Here, we take a different approach. Instead of using general MPC techniques, we extend the Gennaro-Lindell methodology to deal with adaptive corruptions by using a non-malleable conditionally-extractable and equivocable commitment scheme with an associated SPHF family. The new scheme is adaptively secure in the common reference string model in the UC framework under standard complexity assumptions with erasures.

B.1.2 Related work

Commitments. Commitment schemes are one of the most fundamental cryptographic primitives, being used in several cryptographic applications such as zero-knowledge proofs [GMW91] and secure multi-party computation [GMW87]. Even quite practical protocols need them, as already explained above in the public-key registration setting, but also in password-based authenticated key exchange [GL03]. They allow a user to commit a value x into a public value C, such that the latter does not reveal any information about x (the hiding property), but C can be opened later to x only: one cannot change its mind (the binding property). Various additional properties are often required, such as non-malleability, extractability and equivocability. Canetti and Fischlin [CF01] provided an ideal functionality for such a primitive and showed that achieving all these properties at the same time was impossible in the UC plain model. They also provided the first candidate in the CRS model. Damgård and Nielsen [DN02] later proposed another construction of universally composable commitments, that is more efficient for some applications. Since we want to avoid the use of possibly inefficient proofs of relations present in the Damgård-Nielsen construction and given that the Canetti-Fischlin construction is well suited for our purpose of designing an associated smooth hash function, we opted to use the latter as the starting point for our constructions.

PAKE. The password-based setting was first considered by Bellovin and Merritt [BM92] and followed by many proposals. In 2000, Bellare, Pointcheval, and Rogaway [BPR00] as well as Boyko, MacKenzie, and Patel [BMP00] proposed security models and proved variants of the Bellovin and Merritt protocol, under ideal assumptions, such as the random oracle model [BR93]. Soon after, Katz, Ostrovsky, and Yung [KOY01] and Goldreich and Lindell [GL01] proposed the first protocols with a proof of security in the standard model, with the former being based on the decisional Diffie-Hellman assumption and the latter on general assumptions. Later, Gennaro and Lindell [GL03] proposed an abstraction and generalization of the KOY protocol and became the basis of several other variants, including ours in the last section.

B.1.3 Organization of the Paper

In Section B.2, we review the basic primitives needed in this paper. Then, in Section B.3, we describe our first contribution: SPHF families on conjunctions and disjunctions of languages. In Section B.4 we combine that with our second contribution, conditionally-extractable commitments. We focus on the ElGamal-based commitment, since this is enough to build more efficient public-key certification protocols. Finally, in Section B.5, we add equivocability to the commitment, borrowing techniques from Canetti and Fischlin [CF01]. Then, we add the non-malleability property, granted the Cramer-Shoup encryption scheme, which can then be used to build an adaptively-secure PAKE in the UC framework, based on the Gennaro and Lindell [GL03] framework. Due to space restrictions, formal definitions, proofs, and application details were postponed to the appendix.

B.2 Commitments

In the following, we focus on Pedersen commitments, and certification of Schnorr-like public keys, hence, we work in the discrete logarithm setting. As a consequence, to get extractable commitments, we use encryption schemes from the same family: the ElGamal encryption [ElG85b] and the labeled version of the Cramer-Shoup encryption scheme [CS98] (for achieving non-malleability).

Labeled Public-Key Encryption. Labeled encryption [Sho04] is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to its Left-or-Right encryption oracle in the indistinguishability security game (IND-CPA) [GM84, BDJR97], in addition to providing a pair of messages (m_0, m_1), it also has to provide a target label ℓ to obtain the challenge ciphertext c. When chosen-ciphertext security (IND-CCA) is concerned, the adversary is also allowed to query its decryption oracle on any pair (ℓ', c') as long as $\ell' \neq \ell$ or the ciphertext c' does not match the output c of a query to its Left-or-Right encryption oracle whose input includes the label ℓ. For formal security definitions for labeled encryption schemes, please refer to [CHK$^+$05, AP06].

One of the advantages of using labeled encryption, which we exploit in this paper, is that we can easily combine several IND-CCA labeled encryption schemes with the help of a strongly unforgeable one-time signature scheme so that the resulting scheme remains IND-CCA [DK05].

ElGamal and Cramer-Shoup Encryption. We denote by G a cyclic group of prime order q where q is large (n bits), and g a generator for this group. Let $\mathsf{pk} = (g_1, g_2, c = g_1^{x_1} g_2^{x_2}, d =$

$g_1^{y_1} g_2^{y_2}, h = g_1^z, H)$ be the public key of the Cramer-Shoup scheme, where g_1 and g_2 are random group elements, x_1, x_2, y_1, y_2 and z are random scalars in $\mathbb{Z}_q$, and H is a collision-resistant hash function (actually, second-preimage resistance is enough), and $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ the associated private key. Note that (g_1, h) will also be seen as the public key of the ElGamal encryption, with z the associated private key. For the sake of simplicity, we assume in the following that public keys will additionally contain all the global parameters, such as the group G.

If $M \in G$, the multiplicative ElGamal encryption is defined as $\mathsf{EG}^{\times}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r M)$, which can be decrypted by $M = e/u_1^z$. If $M \in \mathbb{Z}_q$, the additive ElGamal encryption is defined as $\mathsf{EG}^{+}_{\mathsf{pk}}(M; r) = (u_1 = g_1^r, e = h^r g^M)$. Note that $\mathsf{EG}^{\times}_{\mathsf{pk}}(g^M; r) = \mathsf{EG}^{+}_{\mathsf{pk}}(M; r)$. It can be decrypted after an additional discrete logarithm computation: M must be small enough. Similarly, if $M \in G$, the multiplicative labeled Cramer-Shoup encryption is defined as $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(M; r) = (u_1, u_2, e, v)$, such that $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r M$, $\theta = H(\ell, u_1, u_2, e)$ and $v = (cd^\theta)^r$. Decryption works as above, with $M = e/u_1^z$, but only if the ciphertext is valid: $v = u_1^{x_1+\theta y_1} u_2^{x_2+\theta y_2}$. If $M \in \mathbb{Z}_q$, its additive encryption $\mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$ is such that $e = h^r g^M$. The following relation holds $\mathsf{CS}^{\times\,\ell}_{\mathsf{pk}}(g^M; r) = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M; r)$. The decryption applies as above if M is small enough.

As already noted, from any Cramer-Shoup ciphertext (u_1, u_2, e, v) of a message M with randomness r, whatever the label ℓ is, one can extract (u_1, e) as an ElGamal ciphertext of the same message M with the same randomness r. This extraction applies independently of the additive or multiplicative version since the decryption works the same for the ElGamal and the Cramer-Shoup ciphertexts, except for the validity check that provides the CCA security level to the Cramer-Shoup encryption scheme, whereas the ElGamal encryption scheme achieves IND-CPA security level only.

Commitments. With a commitment scheme, a player can commit to a secret value x by publishing a commitment $C = \mathsf{com}(x; r)$ with randomness r, in such a way that C reveals nothing about the secret x, which is called the *hiding* property. The player can later open C to reveal x, by publishing x and a decommitment, also referred to as witness, in a publicly verifiable way: the player cannot open C to any other value than x, which is the *binding* property. In many cases, the decommitment consists of the random r itself or some part of it. In this paper, we only consider commitment schemes in the common reference string (CRS) model in which the common parameters, referred to as the CRS, are generated honestly and available to all parties.

Note that an IND-CPA public-key encryption scheme provides such a commitment scheme: the binding property is guaranteed by the uniqueness of the plaintext (perfectly binding), and the hiding property is guaranteed by the IND-CPA security (computationally hiding). In this case, the CRS simply consists of the public-key of the encryption scheme. The Pedersen commitment $C = \mathsf{comPed}(x; r) = g^x h^r$ provides a perfectly hiding, but computationally binding commitment under the intractability of the discrete logarithm of h in basis g.

We now present additional properties that can be satisfied by the commitment. First, we say that a commitment is *extractable* if there exists an efficient algorithm, called an extractor, capable of generating a new set of common parameters (*i.e.*, a new CRS) whose distribution is equivalent to that of an honestly generated CRS and such that it can extract the committed value x from any commitment C. This is of course only possible for computationally hiding commitments, such as encryption schemes: the decryption key is the extraction trapdoor. Second, we say that a commitment is *equivocable* if there exists an efficient algorithm, called an equivocator, capable of generating a new CRS and a commitment with similar distributions to those of the actual scheme and such that the commitment can be opened in different ways. Again, this is possible for computationally binding commitments only, such as the Pedersen commitment: the knowledge

of the discrete logarithm of h in basis g is a trapdoor that allows the opening of a commitment in more than one way. Finally, a *non-malleable* commitment ensures that if an adversary that receives a commitment C of some unknown value x can generate a valid commitment for a related value y, then a simulator could perform as well without seeing the commitment C. A public-key encryption scheme that is IND-CCA provides such a non-malleable commitment [GL03]. For formal security definitions for commitment schemes, please refer to [GL03, DKOS01, CF01].

In the following, we use encryption schemes in order to construct commitments, which immediately implies the hiding, binding and extractable properties, as said above. However, when one uses the additive versions of ElGamal or Cramer-Shoup encryption schemes, extractability (or decryption) is only possible if the committed values (or plaintexts) are small enough, hence our notion of L-extractable commitments (see Section B.4) which will mean that the commitment is extractable if the committed value lies in the language L. More precisely, we will split the value to be committed in small pieces (that lie in the language L), but we will then need to be sure that they actually lie in this language to guarantee extractability. We thus introduce smooth hash functions in order to allow communications if the commitments are valid only.

B.3 Smooth Hash Functions on Conjunctions and Disjunctions of Languages

B.3.1 Smooth Projective Hash Functions.

Projective hash function families were first introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext secure encryption schemes. We here use the definitions of Gennaro and Lindell [GL03], who later showed how to use such families to build secure password-based authenticated key exchange protocols, together with non-malleable commitments. In addition to commitment schemes, we also consider here families of SPHF associated to labeled encryption as done by Canetti *et al.* [CHK$^+$05] and by Abdalla and Pointcheval [AP06].

Let X be the domain of these functions and let L be a certain subset of points of this domain (a language). A key property of these functions is that, for points in L, their values can be computed by using either a *secret* hashing key or a *public* projected key. While the computation using the *secret* hashing key works for all points in the domain X of the hash function, the computation using a *public* projected key only works for points $x \in L$ and requires the knowledge of the witness w to the fact that $x \in L$. A projective hash function family is said to be *smooth* if the value of the function on inputs that are outside the particular subset L of the domain are independent of the projected key. Another important property of these functions is that, given the projected key hp, their output is uniquely defined for points $x \in L$. Moreover, if L is a *hard partitioned subset* of X (*i.e.*, it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$), this output is also *pseudo-random* if one does not know a witness w to the fact that $x \in L$ [GL03]. The interested reader is referred to Appendix B.6 for more formal definitions.

In the particular case of the Gennaro-Lindell scheme [GL03], the subset $L_{\mathsf{pk},m}$ was defined as the set of $\{(c)\}$ such that c is a commitment of m using public parameters pk: there exists r for which $c = \mathsf{com}_{\mathsf{pk}}(m; r)$ where com is the committing algorithm of the commitment scheme. In the case of the CHKLM scheme [CHK$^+$05], the subset $L_{\mathsf{pk},(\ell,m)}$ was defined as the set of $\{(c)\}$ such that c is an encryption of m with label ℓ, under the public key pk: there exists r for which $c = \mathcal{E}^{\ell}_{pk}(m; r)$ where $\mathcal{E}$ is the encryption algorithm of the labeled encryption scheme. In the case of a standard encryption scheme, the label is simply omitted. The interested reader is referred to [GL03, CHK$^+$05, AP06] for more details.

Languages. Since we want to use more general languages, we need more detailed notations.

Let **LPKE** be a labeled encryption scheme with public key pk. Let X be the range of the encryption algorithm. Here are three useful examples of languages L in X:

- the valid ciphertexts c of m under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m)} = \{c | \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$;
- the valid ciphertexts c of m_1 or m_2 under pk (that is, a disjunction of two versions of the former languages), $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1 \vee m_2)} = L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_1)} \cup L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,m_2)}$;
- the valid ciphertexts c under pk, $L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,*)} = \{c | \exists m\ \exists r\ c = \mathcal{E}^{\ell}_{\mathsf{pk}}(m;r)\}$.

If the encryption scheme is IND-CPA, the first two are hard partitioned subsets of X. The last one can also be a hard partitioned subset in some cases: for the Cramer-Shoup encryption, $L \subsetneq X = G^4$ and, in order to distinguish a valid ciphertext from an invalid one, one has to break the DDH problem. However, for the ElGamal encryption scheme, all the ciphertexts are valid, hence $L = X = G^2$.

More complex languages can be defined, with disjunctions as above, or conjunctions: the pairs of ciphertexts (a, b) such that $a \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee1)}$ and $b \in L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee3)}$. This set can be obtained by $(L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,0\vee1))} \times X) \cap (X \times L_{(\mathbf{LPKE},\mathsf{pk}),(\ell,2\vee3)})$.

Likewise, we can define more general languages based on other primitives such as commitment schemes. The definition would be similar to the one above, with pk playing the role of the common parameters, $\mathcal{E}_{\mathsf{pk}}$ playing the role of the committing algorithm, (m, ℓ) playing the role of the input message, and c playing the role of the commitment.

More generally, in the following, we denote the language by the generic notation $L_{(\mathbf{Sch},\rho),aux}$ where aux denotes all the parameters useful to characterize the language (such as the label used, or a plaintext), ρ denotes the public parameters such as a public key pk, and **Sch** denotes the primitive used to define the language, such as an encryption scheme **LPKE** or a commitment scheme **Com**. When there is no ambiguity, the associated primitive **Sch** will be omitted.

We now present new constructions of SPHF to deal with more complex languages, such as disjunctions and conjunctions of any languages. The constructions are presented for two languages but can be easily extended to any polynomial number of languages. We then discuss about possible information leakage at the end of this section. The properties of correctness, smoothness and pseudo-randomness are easily verified by these new smooth hash systems. Due to the lack of space, the formal proofs can be found in Appendix B.7.

B.3.2 Conjunction of two Generic Smooth Hashes

Let us consider an encryption or commitment scheme defined by public parameters and a public key aggregated in ρ. X is the range of the elements we want to study (ciphertexts, tuples of ciphertexts, commitments, etc), and $L_1 = L_{1,\rho,aux}$ and $L_2 = L_{2,\rho,aux}$ are hard partitioned subsets of X, which specify the expected properties (valid ciphertexts, ciphertexts of a specific plaintext, etc). We consider situations where X possesses a group structure, which is the case if we consider ciphertexts or tuples of ciphertexts from an homomorphic encryption scheme. We thus denote by $\oplus$ the commutative law of the group (and by $\ominus$ the opposite operation, such that $c \oplus a \ominus a = c$).

We assume to be given two smooth hash systems SHS_1 and SHS_2, on the sets corresponding to the languages L_1 and L_2: $\mathsf{SHS}_i = \{\mathsf{HashKG}_i, \mathsf{ProjKG}_i, \mathsf{Hash}_i, \mathsf{ProjHash}_i\}$. Here, HashKG_i and ProjKG_i denote the hashing key and the projected key generators, and Hash_i and $\mathsf{ProjHash}_i$ the algorithms that compute the hash function using hk_i and hp_i respectively.

Let c be an element of X, and r_1 and r_2 two elements chosen at random. We denote by $\mathsf{hk}_1 = \mathsf{HashKG}_1(\rho, aux, r_1)$, $\mathsf{hk}_2 = \mathsf{HashKG}_2(\rho, aux, r_2)$, $\mathsf{hp}_1 = \mathsf{ProjKG}_1(\mathsf{hk}_1; \rho, aux, c)$, and

$\mathsf{hp}_2 = \mathsf{ProjKG}_2(\mathsf{hk}_2; \rho, aux, c)$ the keys. A smooth hash system for the language $L = L_1 \cap L_2$ is then defined as follows, if $c \in L_1 \cap L_2$ and w_i is a witness that $c \in L_i$, for $i = 1, 2$:

$$\begin{aligned}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) \\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c) \\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; (w_1, w_2)) &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w_1) \oplus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w_2)
\end{aligned}$$

B.3.3 Disjunction of two Generic Smooth Hashes

Let L_1 and L_2 be two languages as described above. We assume to be given two smooth hash systems SHS_1 and SHS_2 with respect to these languages. We define $L = L_1 \cup L_2$ and construct a smooth projective hash function for this language as follows:

$$\begin{aligned}
\mathsf{HashKG}_L(\rho, aux, r = r_1 \| r_2) &= \mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2) \\
\mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, c)) \\
\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, c) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, c) \\
\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, c; w) &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, c; w) \quad \text{if } c \in L_1 \\
&\quad \text{or } \mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, c; w) \text{ if } c \in L_2
\end{aligned}$$

where w is a witness of $c \in L_i$ for $i \in \{1, 2\}$. Then $\mathsf{ProjHash}_i(\mathsf{hp}_i; \rho, aux, c; w) = \mathsf{Hash}_i(\mathsf{hk}_i; \rho, aux, c)$. The player in charge of computing this value is supposed to know the witness, and in particular the language which c belongs to (and thus the index i).

B.3.4 Uniformity and Independence

In the above definition of SPHF (contrarily to the original Cramer-Shoup [CS02] definition), the value of the projected key formally depends on the ciphertext/commitment c. However, in some cases, one may not want to reveal any information about this dependency. In fact, in certain cases such as in the construction of a SPHF for equivocable and extractable commitments in Section B.5, one may not even want to leak any information about the auxiliary elements aux. When no information is revealed about aux, it means that the details about the exact language will be concealed.

We thus add a notion similar to the smoothness, but for the projected key: the projected key may or may not depend on c (and aux), but its distribution does not: Let us denote by $D_{\rho,aux,c}$ the distribution $\{\mathsf{hp} \mid \mathsf{hk} = \mathsf{HashKG}_L(\rho, aux, r) \text{ and } \mathsf{hp} = \mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, c)\}$, on the projected keys. If, for any $c, c' \in X$, $D_{\rho,aux,c'}$ and $D_{\rho,aux,c}$ are indistinguishable, then we say that the smooth hash system has the *1-uniformity* property. If, for any $c, c' \in X$, and any auxiliary elements aux, aux', $D_{\rho,aux',c'}$ and $D_{\rho,aux,c}$ are indistinguishable, we name it *2-uniformity* property.

More than indistinguishability of distributions, the actual projected key hp may not depend at all on c, as in the Cramer and Shoup's definition. Then, we say that the smooth hash system guarantees *1-independence* (resp. *2-independence* if it does not depend on aux either). Note that the latter independence notions immediately imply the respective uniformity notions.

As an example, the smooth hash system associated with the ElGamal cryptosystem (see Section B.4.1) guarantees 2-independence. On the other hand, the analogous system associated with the Cramer-Shoup encryption (see Appendix B.9.1) guarantees 2-uniformity only. For smooth hash systems combinations, one can note that in the case of disjunctions, one can get, at best, the uniformity property, since hash computations on the commitment are needed for generating the projected key. Furthermore, this is satisfied under the condition that the two

underlying smooth hash systems already satisfy this property (see Appendix B.7 for more details and proofs).

Finally, one should note that, in the case of disjunction, the view of the projected hash value could leak some information about the sub-language in which the input lies, if an adversary sends a fake hp_Δ. The adversary could indeed check whether $\mathsf{ProjHash}_L(\mathsf{hp};\rho,aux,c;w)$ equals $\mathsf{Hash}_1(\mathsf{hk}_1;\rho,aux,c)$ or $\mathsf{hp}_\Delta \ominus \mathsf{Hash}_2(\mathsf{hk}_2;\rho,aux,c)$. But first, it does not contradict any security notion for smooth hash systems; second, in all the applications below, the projected hash value is never revealed; and third, in the extractable commitments below, because of the global conjunction of the languages, an exponential exhaustive search would be needed to exploit this information, even if the committed value is a low-entropy one.

B.4 A Conditionally Extractable Commitment

B.4.1 ElGamal Commitment and Associated Smooth Hash

The ElGamal commitment is realized in the common reference string model, where the CRS ρ contains (G,pk), as defined in Section B.2, for the ElGamal encryption scheme. In practice, sk should not be known by anybody, but in the security analysis, sk will be the extraction trapdoor. Let the input of the committing algorithm be a scalar $M \in \mathbb{Z}_q$. The commitment algorithm consists of choosing a random r and computing the following ElGamal encryption under random r: $C = \mathsf{EG}^+_{\mathsf{pk}}(M,r) = (u_1 = g_1^r, e = h^r g^M)$.

The smooth projective hashing, associated with this commitment scheme and the language $L = L_{(\mathbf{EG}^+,\rho),M} \subset X = G^2$ of the additive ElGamal ciphertexts C of M under the global parameters and public key defined by ρ, is the family based on the underlying ElGamal encryption scheme, as defined in [GL03]:

$$\begin{aligned}
&\mathsf{HashKG}((\mathbf{EG}^+,\rho),M) = \mathsf{hk} = (\gamma_1 \grave{u}, \gamma_3 \grave{u}) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q\\
&\mathsf{Hash}(\mathsf{hk};(\mathbf{EG}^+,\rho),M,C) = (u_1)^{\gamma_1 \grave{u}} (e g^{-M})^{\gamma_3 \grave{u}}\\
&\mathsf{ProjKG}(\mathsf{hk};(\mathbf{EG}^+,\rho),M,C) = \mathsf{hp} = (g_1)^{\gamma_1 \grave{u}} (h)^{\gamma_3 \grave{u}}\\
&\mathsf{ProjHash}(\mathsf{hp};(\mathbf{EG}^+,\rho),M,C;r) = (\mathsf{hp})^r
\end{aligned}$$

First, under the Decisional Diffie-Hellman problem (semantic security of the ElGamal encryption scheme), L is a hard partitioned subset of $X = G^2$. Then, for $C = \mathsf{EG}^+_{\mathsf{pk}}(M,r)$, and thus with the witness r, the algorithms are defined as above using the same notations as in [GL03].

B.4.2 L-extractable Commitments

Note that the value g^M would be easily extractable from this commitment (seen as the multiplicative ElGamal encryption). However, one can extract M itself (the actual committed value) only if its size is small enough so that it can be found as a solution to the discrete logarithm problem. In order to obtain "extractability" (up to a certain point, see below), one should rather commit to it in a bit-by-bit way.

Let us denote $M \in \mathbb{Z}_q$ by $\sum_{i=1}^m M_i \cdot 2^{i-1}$, where $m \leq n$. Its commitment is $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1,\ldots,b_m)$, where $b_i = \mathsf{EG}^+_{\mathsf{pk}}(M_i \cdot 2^{i-1}, r_i) = (u_{1,i} = {g_1}^{r_i}, e_i = h^{r_i} g^{M_i \cdot 2^{i-1}})$, for $i = 1,\ldots,m$. The homomorphic property of the encryption scheme allows to obtain, from this tuple, the above simple commitment of M

$$C = \mathsf{EG}^+_{\mathsf{pk}}(M,r) = (u_1,e) = (\textstyle\prod u_{1,i}, \prod e_i) = \prod b_i, \text{ for } r = \sum r_i.$$

We now precise what we mean by "extractability": Here, the commitment will be extractable if the messages M_i are bits (or at least small enough), but we cannot ensure that it will be extractable otherwise. More generally, this leads to a new notion of *$L-$extractable commitments*, which means that we allow the primitive not to be extractable if the message does not belong

to a certain language L (*e.g.* the language of encryptions of 0 or 1), which is informally the language of all commitments valid and "of good shape", and is included into the set X of all commitments.

Smooth Hash Functions. For the above protocol, we need a smooth hash system on the language $L = L_1 \cap L_2$, where $L_1 = \{(b_1, \ldots, b_m) \mid \forall i, b_i \in L_{(\mathbf{EG}^+,\rho),0\vee1}\}$, $L_2 = \{(b_1, \ldots, b_m) \mid C = \prod_i b_i \in L_{(\mathbf{EG}^\times,\rho),g^M}\}$, to within a factor (corresponding to the offest 2^{i-1}) with

$$L_{(\mathbf{EG}^+,\rho),0\vee1} = L_{(\mathbf{EG}^+,\rho),0} \cup L_{(\mathbf{EG}^+,\rho),1} \qquad L_{(\mathbf{EG}^+,\rho),0} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(0,r)\}$$
$$L_{(\mathbf{EG}^\times,\rho),g^M} = \{C \mid \exists r\ C = \mathsf{EG}^\times_{\mathsf{pk}}(g^M,r)\} \quad L_{(\mathbf{EG}^+,\rho),1} = \{C \mid \exists r\ C = \mathsf{EG}^+_{\mathsf{pk}}(1,r)\}$$

It is easy to see that this boils down to constructing a smooth hash system corresponding to a conjunction and disjunction of languages, as presented in the previous section.

B.4.3 Certification of Public Keys

Description. A classical application of extractable commitments is in the certification of public keys (when we want to be sure that a person joining the system actually knows the associated private key). Suppose that a user U owns a pair of secret and public keys, and would like to have the public key certified by the authority. A natural property is that the authority will not certify this public key unless it is sure that the user really owns the related private key, which is usually ensured by a zero-knowledge proof of knowledge: the user knows the private key if a successful extractor exists.

Here we present a construction that possesses the same property without requiring any explicit proof of knowledge, furthermore in a concurrent way since there is no need of any rewinding:

- First, the user sends his public key g^M, along with a bit-by-bit L-extractable commitment of the private key M, i.e. a tuple $\mathsf{comEG}_{\mathsf{pk}}(M) = (b_1, \ldots, b_m)$ as described above, from which one can derive $C = \prod b_i = \mathsf{EG}^+_{\mathsf{pk}}(M,r) = \mathsf{EG}^\times_{\mathsf{pk}}(g^M,r)$.

- We define the smooth hash system related to the language $L_1 \cap L_2$, where $L_1 = \cap_i L_{1,i}$, with $L_{1,i}$ the language of the tuples where the i-th component b_i is an encryption of 0 or 1, and L_2 is the language of the tuples where the derived $C = \prod b_i$ is an encryption of the public key g^M (under the multiplicative ElGamal, as in Section B.4.1).

  Note that when the tuple $(b_1, \ldots, b_m)$ lies in $L_1 \cap L_2$, it really corresponds to an extractable commitment of the private key M associated to the public key g^M: each b_i encrypts a bit, and can thus be decrypted, which provides the i-th bit of M.

- The authority computes a hash key hk, the corresponding projected key hp on $(b_1, \ldots, b_m)$ and the related hash value Hash on $(b_1, \ldots, b_m)$. It sends hp to U along with $\mathsf{Cert} \oplus \mathsf{Hash}$, where Cert is the expected certificate. Note that if Hash is not large enough, a pseudo-random generator can be used to expand it.

- The user is then able to recover his certificate if and only if he can compute Hash: this value can be computed with the algorithm $\mathsf{ProjHash}$ on $(b_1, \ldots, b_m)$, from hp. But it also requires a witness w proving that the tuple $(b_1, \ldots, b_m)$ lies in $L_1 \cap L_2$.

With the properties of the smooth hash system, if the user correctly computed the commitment, he knows the witness w, and can get the same mask Hash to extract the certificate. If the user cheated, the smoothness property makes Hash perfectly unpredictable: no information is leaked about the certificate.

Security Analysis. Let us outline the security proof of the above protocol. First, the security model is the following: no one can obtain a certificate on a public key if it does not know the associated private key (that is, if no simulator can extract the private key). In other words, the adversary wins if it is able to output (g^M, Cert) and no simulator can produce M.

The formal attack game can thus be described as follows: the adversary $\mathcal{A}$ interacts several times with the authority, by sending public keys and commitments, and asks for the corresponding certificates. It then outputs a pair (g^M, Cert) and wins if no simulator is able to extract M from the transcript.

The simulator works as follows: it is given access to a certification (signing) oracle, and generates a pair of public and private keys $(\mathsf{sk}, \mathsf{pk})$ for the ElGamal encryption. The public key is set as the CRS that defines the commitment scheme. The private key will thus be the extraction trapdoor.

When the simulator receives a certification request, with a public key and a commitment, it first tries to extract the associated private key, granted the extraction trapdoor. In case of success, the simulator asks the signing oracle to provide it with the corresponding certificate on the public key, and complete the process as described in the protocol. However, extraction may fail if the commitments are not well constructed (not in $L_1 \cap L_2$). In such a case, the simulator sends back a random bit-string of appropriate length. In case of successful extraction, the answer received by the user is exactly the expected one. In case of failure, it is perfectly indistinguishable too since the smoothness property of the hash function would make a perfectly random mask Hash (since the input is not in the language).

After several interactions, $\mathcal{A}$ outputs a pair (g^M, Cert), which is forwarded by the simulator. Either g^M has been queried to the signing oracle, which means that the extraction had succeeded, the simulator knows M and the adversary did not win the attack game, or this is a valid signature on a new message: existential forgery under chosen-message attack.

B.5 A Conditionally Extractable Equivocable Commitment

In this section, we enhance the previous commitment schemes with equivocability, which is not a trivial task when one wants to keep the extraction property. Note that we first build a malleable extractable and equivocable commitment using the ElGamal-based commitment (see Section B.4.1), but one can address the non-malleability property by simply building the commitment upon the Cramer-Shoup encryption scheme. All the details of this extension are given in Appendix B.9. In the following, if b is a bit, we denote its complement by $\overline{b}$ (i.e., $\overline{b} = 1 - b$). We furthermore denote by $x[i]$ the i^{th} bit of the bit-string x.

B.5.1 Equivocability

Commitments that are both extractable and equivocable seem to be very difficult to obtain. Canetti and Fischlin [CF01] proposed a solution but for one bit only. Damgård and Nielsen [DN02] proposed later another construction. But for efficiency reasons, in our specific context, we extend the former proposal. In this section, we thus enhance our previous commitment (that is already L-extractable) to make it equivocable, using the Canetti and Fischlin's approach. Section B.5.3 will then apply a non-malleable variant of our new commitment together with the associated smooth hash function family in order to build a password-authenticated key exchange protocol with adaptive security in the UC framework [Can01]. The resulting protocol is reasonably efficient and, in particular, more efficient than the protocol by Barak *et al.* [BCL$^+$05], which to our knowledge is the only one achieving the same level of security in the standard model.

Description of the Commitment. Our commitment scheme is a natural extension of Canetti-Fischlin commitment scheme [CF01], in a bit-by-bit way. It indeed uses the ElGamal public-key encryption scheme, for each bit of the bit-string. Let $(y_1, \ldots, y_m)$ be random elements in G. This commitment is realized in the common reference string model, the CRS ρ contains (G, pk), where pk is an ElGamal public key and the private key is unknown to anybody, except to the commitment extractor. It also includes this tuple $(y_1, \ldots, y_m)$, for which the discrete logarithms in basis g are unknown to anybody, except to the commitment equivocator. Let the input of the committing algorithm be a bit-string $\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$. The algorithm works as follows:

- For $i = 1, \ldots, m$, it chooses a random value $x_{i,\pi_i} = \sum_{j=1}^{n} x_{i,\pi_i}[j] \cdot 2^{j-1}$ and sets $x_{i,\overline{\pi}_i} = 0$.
- For $i = 1, \ldots, m$, the algorithm commits to π_i, using the random x_{i,π_i}: $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}} y_i^{\pi_i}$ and defining $\mathbf{a} = (a_1, \ldots, a_m)$.
- For $i = 1, \ldots, m$, it computes the ElGamal commitments (see the previous section) of $x_{i,\delta}$, for $\delta = 0, 1$: $(\mathbf{b}_{i,\delta} = (b_{i,\delta}[j])_j = \mathsf{comEG}_{\mathsf{pk}}(x_{i,\delta})$, where $b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j])$. One can directly extract from the computation of the $b_{i,\delta}[j]$ an encryption $B_{i,\delta}$ of $x_{i,\delta}$: $B_{i,\delta} = \prod_j b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}, r_{i,\delta})$, where $r_{i,\delta}$ is the sum of the random coins $r_{i,\delta}[j]$.

The entire random string for this commitment is (where n is the bit-length of the prime order q of the group G) $R = (x_{1,\pi_1}, (r_{1,0}[1], r_{1,1}[1], \ldots, r_{1,0}[n], r_{1,1}[n]), \ldots, x_{m,\pi_m}, (r_{m,0}[1], \ldots, r_{m,1}[n]))$. From which, all the values $r_{i,\overline{\pi}_i}[j]$ can be erased, letting the opening data (witness of the committed value) become limited to $\mathsf{w} = (x_{1,\pi_1}, (r_{1,\pi_1}[1], \ldots, r_{1,\pi_1}[n]), \ldots, x_{m,\pi_m}, (r_{m,\pi_m}[1], \ldots, r_{m,\pi_m}[n]))$. The output of the committing algorithm, of the bit-string π, using the random R, is $\mathsf{com}_\rho(\pi; R) = (\mathbf{a}, \mathbf{b})$, where $\mathbf{a} = (a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}))_i, \mathbf{b} = (b_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]))_{i,\delta,j}$.

Opening. In order to open this commitment to π, the above witness w (with the value π) is indeed enough: one can build again, for all i and j, $b_{i,\pi_i}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\pi_i}[j] \cdot 2^{j-1}, r_{i,\pi_i}[j])$, and check them with $\mathbf{b}$. One can then also compute again all the $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i})$, and check them with $\mathbf{a}$. The erased random elements would help to check the encryptions of zeroes, what we do not want, since the equivocability property will exploit that.

Properties. Let us briefly check the security properties, which are formally proven in Appendix B.8. First, because of the perfectly hiding property of the Pedersen commitment, unless some information is leaked about the $x_{i,\delta}[j]$'s, no information is leaked about the π_i's. And granted the semantic security of the ElGamal encryption scheme, the former privacy is guaranteed. Since the Pedersen commitment is (computationally) binding, the a_i's cannot be opened in two ways, but only one pair (π_i, x_{i,π_i}) is possible. Let us now consider the new extended properties:

- (conditional) extractability is provided by the bit-by-bit encryption. With the decryption key sk, one can decrypt all the $b_{i,\delta}[j]$, and get the $x_{i,\delta}$ (unless the ciphertexts contain values different from 0 and 1, which will be one condition for extractability). Then, one can check, for $i = 1, \ldots, m$, whether $a_i = \mathsf{comPed}(0, x_{i,0})$ or $a_i = \mathsf{comPed}(1, x_{i,1})$, which provides π_i (unless none of the equalities is satisfied, which will be another condition for extractability).
- equivocability is possible using the Pedersen commitment trapdoor. Instead of taking a random x_{i,π_i} and then $x_{i,\overline{\pi}_i} = 0$, which specifies π_i as the committed bit, one takes a random $x_{i,0}$, computes $a_i = \mathsf{comPed}(0, x_{i,0})$, but also extracts $x_{i,1}$ so that $a_i = \mathsf{comPed}(1, x_{i,1})$ too (which is possible with the knowledge of discrete logarithm of y_i in basis g, the trapdoor). The rest of the commitment procedure remains the same, but now, one can open any

bit-string for π, using the appropriate x_{i,π_i} and the corresponding random elements (the simulator did not erase).

B.5.2 The Associated Smooth Projective Hash Function

As noticed above, our new commitment scheme is conditionally extractable (one can recover the $x_{i,\delta}$'s, and then the committed value π), under the conditions that all the ElGamal ciphertexts encrypt either 0 or 1, and the a_i is a commitment of either 0 or 1, with random $x_{i,0}$ or $x_{i,1}$.

As before, one wants to make the two hash values (direct computation and the one from the projected key) be the same if the two parties use the same input π and perfectly independent if they use different inputs (smoothness). One furthermore wants to control that each a_i is actually a Pedersen commitment of π_i using the encrypted random x_{i,π_i}, and thus $g^{x_{i,\pi_i}} = a_i/y_i^{\pi_i}$: the extracted x_{i,π_i} is really the private key M related to a given public key g^M that is $a_i/y_i^{\pi_i}$ in our case. Using the same notations as in Section B.4.1, we want to define a smooth hash system showing that, for all i, δ, j, $b_{i,\delta}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee1}$ and, for all i, $B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),(a_i/y_i{}^{\pi_i})}$, where $B_{i,\pi_i} = \prod_j b_{i,\pi_i}[j]$.

Combinations of these smooth hashes. Let C be the above commitment of π using randomness R as defined in Section B.5.1. We now precise the language $L_{\rho,\pi}$, consisting informally of all the valid commitments "of good shape":

$$L_{\rho,\pi} = \left\{ C \;\middle|\; \begin{array}{ll} \exists R \text{ s. t. } C = \mathsf{com}_\rho(\pi, R) & \text{and } \forall i\ \forall j \quad b_{i,\pi_i}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee1} \\ & \text{and } \forall i \quad B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),a_i/y_i{}^{\pi_i}} \end{array} \right\}$$

The smooth hash system for this language relies on the smooth hash systems described previously, using the generic construction for conjunctions and disjunctions as described in Section B.3. The precise definition of this language (which is constructed from conjunctions and disjunctions of simple languages) can be found in Appendix B.9, omitting the labels and replacing the Cramer-Shoup encryption $\mathbf{CS}^+$ by the ElGamal one $\mathbf{EG}^+$.

Properties: Uniformity and Independence. With a non-malleable variant of such a commitment and smooth hash function, it is possible to improve the establishment of a secure channel between two players, from the one presented Section B.4.3. More precisely, two parties can agree on a common key if they both share a common (low entropy) password π. However, a more involved protocol than the one proposed in Section B.4.3 is needed to achieve all the required properties of a password-authenticated key exchange protocol, as it will be explained in Section B.5.3 and proven in Appendix B.10.

Nevertheless, there may seem to be a leakage of information because of the language that depends on the input π: the projected key hp seems to contain some information about π, that can be used in another execution by an adversary. Hence the independence and uniformity notions presented Section B.3.4, which ensure that hp does not contain any information about π. Proofs of these properties can be found in Appendix B.9.

Estimation of the Complexity. Globally, each operation (commitment, projected key, hashing and projected hashing) requires $\mathcal{O}(mn)$ exponentiations in G, with small constants (at most 16).

B.5.3 UC-Secure PAKE with Adaptive Security

The primitive presented above, but using the Cramer-Shoup encryption scheme (as described in Section B.9) is a non-malleable conditionally extractable and equivocable commitment. We now sketch how to use this new primitive in order to construct the first efficient adaptively-secure

password-authenticated key exchange protocol in the UC framework with erasures. For lack of space, all the details can be found in Appendix B.10. The passwords are not known at the beginning of the simulation: $\mathcal{S}$ will manage to correct the errors (thanks to the equivocability) but without erasures there would remain clues on how the computations were held, which would give indications on the passwords used.

Our protocol is based on that of Gennaro and Lindell [GL03]. At a high level, the players in the KOY/GL protocol exchange CCA-secure encryptions of the password, under the public-key found in the common reference string, which are essentially commitments of the password. Then, they compute the session key by combining smooth projective hashes of the two password/ciphertext pairs. The security of this protocol relies on the properties of smoothness and pseudo-randomness of the smooth projective hash function. But as noted by Canetti *et al* in [CHK$^+$05], the KOY/GL protocol is not known to achieve UC security: the main issue is that the ideal-model simulator must be able to extract the password used by the adversary before playing, which is impossible if the simulator is the initiator (on behalf of the client), leading to such situation in which the simulator is stuck with an incorrect ciphertext and will not be able to predict the value of the session key.

To overcome this problem, the authors of [CHK$^+$05] made the client send a pre-flow which also contains an encryption of the password. The server then sends its own encryption, and finally the client sends another encryption, as well as a zero-knowledge proof showing that both ciphertexts are consistent and encrypt the same password. This time the simulator, playing as the client or the server, is able to use the correct password, recovered from the encrypted value sent earlier by the other party. The pre-flow is never used in the remaining of the protocol, hence the simulator can send a fake one, and simulate the zero-knowledge proof.

Unfortunately, the modification above does not seem to work when dealing with adaptive adversaries, which is the case in which we are interested. This is because the simulator cannot correctly open the commitment when the adversary corrupts the client after the pre-flow has been sent. A similar remark applies to the case in which the server gets corrupted after sending its first message. As a result, in addition to being extractable, the commitment scheme also needs to be equivocable for the simulator to be able to provide a consistent view to the adversary. Since the use of the equivocable and extractable commitment schemes also seems to solve the problem of proving the original Gennaro-Lindell protocol secure in the UC model, we opted to use that protocol as the starting point of our protocol.

These remarks are indeed enough (along with minor modifications) to obtain adaptive security. Thus, our solution essentially consists in using our non-malleable extractable and equivocable commitment scheme in the Gennaro-Lindell protocol when computing the first two flows. As presented in the previous subsections, extractability may be conditional: We include this condition in the language of the smooth hash function (note that the projected keys sent do not leak any information about the password). Additional technical modifications were also needed to make things work and can be found in Appendix B.10.

Acknowlegments

This work was supported in part by the French ANR-07-SESU-008-01 PAMPA Project, and the European ECRYPT Project.

B.6 Appendix: Formal Definitions for Smooth Projective Hash Functions

As defined in [GL03], a family of smooth projective hash functions, for a language $L_{\mathsf{pk},aux} \subset X$, onto the set G, based on a labeled encryption scheme with public key pk or on a commitment scheme with public parameters pk consists of four algorithms and is denoted by $\mathbf{HASH}(\mathsf{pk}) = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$. Note that X is the range of either the encryption or commitment algorithm.

The probabilistic key-generation algorithm produces hash keys via $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{pk}, aux)$. The key projection algorithm produces projected hash keys via $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c)$, where c is either a ciphertext or a commitment in X. The hashing algorithm Hash computes, on $c \in X$, the hash value $g = \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c) \in G$, using the hash key hk. Finally, the projected hashing algorithm $\mathsf{ProjHash}$ computes, on $c \in X$, the hash value $g = \mathsf{ProjHash}(\mathsf{hp}; \mathsf{pk}, aux, c; w) \in G$, using the projected hash key hp and a witness w of the fact that $c \in L_{\mathsf{pk},aux}$.

We now recall the three properties of a smooth hash system.

Correctness. Let $c \in L_{\mathsf{pk},aux}$ and w a witness of this membership. Then, for all hash keys and projected hash keys $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{pk}, aux)$ and $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c)$, then $\mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c) = \mathsf{ProjHash}(\mathsf{hp}; \mathsf{pk}, aux, c; w)$.

Smoothness. For every c which is *not* in $L_{\mathsf{pk},aux}$, the hash value $g = \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c)$ is statistically close to uniform and independent of the values hp, pk, aux and c: for uniformly-chosen hash key hk, the two distributions are statistically indistinguishable:

$$\{\mathsf{pk}, aux, c, \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c), \quad g = \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c)\}$$
$$\{\mathsf{pk}, aux, c, \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c), \quad g \xleftarrow{\$} G\}$$

Pseudorandomness. If $c \in L_{\mathsf{pk},aux}$, then without a witness w of this membership, the hash value $g = \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c)\}$ is computationally indistinguishable from random: for uniformly-chosen hash key hk, the following two distributions are computationally indistinguishable:

$$\{\mathsf{pk}, aux, c, \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c), \quad g = \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, c)\}$$
$$\{\mathsf{pk}, aux, c, \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, c), \quad g \xleftarrow{\$} G\}$$

Let $L_{\mathsf{pk},aux}$ be such that it is hard to distinguish a random element in $L_{\mathsf{pk},aux}$ from a random element not in $L_{\mathsf{pk},aux}$. Gennaro and Lindell formalized the latter property by showing in [GL03] that the two following experiments are indistinguishable:

`Expt-Hash`(D): Let D be a adversary that is given access to two oracles: Ω and `Hash`. The first oracle receives an empty input and returns $x \in L_{\mathsf{pk},aux}$ chosen according to the distribution of $L_{\mathsf{pk},aux}$. The Hash oracle receives an input x. If x was not previously outputted by Ω, it outputs nothing. Otherwise, it chooses a key hk and returns the pair $(\mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, x), \mathsf{Hash}(\mathsf{hk}; \mathsf{pk}, aux, x))$. The output of the experiment is whatever M outputs.

`Expt-Unif`(D): This experiment is defined exactly as above except that the `Hash` oracle is replaced by the following `Unif` oracle. On input x, if x was not previously outputted by Ω, it outputs nothing. Otherwise, it chooses a key hk and a random element g and returns the pair $(\mathsf{ProjKG}(\mathsf{hk}; \mathsf{pk}, aux, x), g)$. The output of the experiment is whatever M outputs.

In the case where the language $L_{\mathsf{pk},aux}$ is associated with a labeled encryption scheme, we rename the oracle Ω to `Enc`. In the case where the language $L_{\mathsf{pk},aux}$ is associated with a commitment scheme, we rename the oracle Ω to `Commit`.

B.7 Appendix: Proofs for SPHF Constructions in Section B.3

In this appendix, we prove the properties of the smooth projective hash functions on conjunctions and disjunctions of languages.

B.7.1 Disjunction

We first deal with $L = L_1 \cup L_2$, and study the additional information provided by the projected key, which contains hp_1 and hp_2, but also $\mathsf{hp}_\Delta = \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x)$. Since both SHS_1 and SHS_2 are smooth projective hash functions, the pseudo-randomness property for each of them (or even the smoothness, if x does not lie in one of the languages) guarantees that the pairs $(\mathsf{hp}_i, \mathsf{Hash}_i(\mathsf{hk}_i; \rho, aux, x))$ are (statistically or computationally) indistinguishable from (hp_i, g_i). As a consequence, one easily gets that the tuple $(\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x), \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x))$ is (statistically or computationally) indistinguishable from $(\mathsf{hp}_1, \mathsf{hp}_2, g_1, g_2)$, where g_1 and g_2 are independent. This *a fortiori* implies that the tuple $(\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x))$ is (statistically or computationally) indistinguishable from $(\mathsf{hp}_1, \mathsf{hp}_2, g)$: the element hp_Δ does not provide any additional information.

Efficient Hashing from Key. Given any element $x \in X$ and a key hk, it is possible to efficiently compute $\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x)$.

Proof: This follows from the efficient hashings of the two underlying smooth projective hash functions, and namely SHS_1, since $\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x) = \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x)$. ∎

Efficient Hashing from Projected Key. Given an element $x \in L$, a witness w of this membership, and the projected key $\mathsf{hp} = \mathsf{ProjKG}_L(\mathsf{hk}; \rho, aux, x)$, it is possible to efficiently compute the projected hash value $\mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, x, w) = \mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x)$.

Proof: If $x \in L_1$, then,

$$\begin{aligned}\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x)\\ &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, x, w) = \mathsf{ProjHash}_L(\mathsf{hp}; \rho, aux, x, w),\end{aligned}$$

which can be computed efficiently since SHS_1 is a smooth projective hash function. If $x \in L_2$, then,

$$\begin{aligned}\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x) &= \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x)\\ &= \mathsf{ProjHash}_1(\mathsf{hp}_1; \rho, aux, x, w) = \mathsf{hp}_\Delta \ominus \mathsf{ProjHash}_2(\mathsf{hp}_2; \rho, aux, x, w),\end{aligned}$$

which can be computed efficiently since SHS_2 is a smooth projective hash function. ∎

Smoothness. For each element $x \in X \setminus L$, $\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x)$ is uniformly distributed, given the projected key.

Proof: Consider $x \notin L$. Then, $x \notin L_1$ and $x \notin L_2$. If $\mathsf{hk} = (\mathsf{hk}_1, \mathsf{hk}_2)$ is a random key, and $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta)$ the corresponding projected key, then the above analysis showed the statistical indistinguishability between the tuples $(\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x), \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x))$ and $(\mathsf{hp}_1, \mathsf{hp}_2, g_1, g_2)$, where g_1 and g_2 are random and independent elements. This means the statistical indistinguishability between the tuples $(\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_\Delta, \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x))$ and $(\mathsf{hp}_1, \mathsf{hp}_2, g_1, g_2)$. As a consequence, the tuple $(\mathsf{hp}, \mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x))$ is statistically indistinguishable from (hp, g), which is the definition of the smoothness for the new system. ∎

Pseudo-Randomness. For each element $x \in L$, the value $\mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x)$ is computationally indistinguishable from uniform, given the projected key.

Proof: Exactly the same analysis as in the previous paragraph can be done, but with computational indistinguishability when $x \in L_1$ or $x \in L_2$. Hence one gets that the tuple $(\mathsf{hp}, \mathsf{Hash}_L(\mathsf{hk}; \rho, aux, x))$ is computationally indistinguishable from (hp, g), which is the definition of the pseudo-randomness for the new system. ▮

B.7.2 Conjunction

The case of $L = L_1 \cap L_2$, can be dealt as above: $(\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x) \oplus \mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x))$ is statistically (if $x \notin L_1$ or $x \notin L_2$) or computationally (if $x \in L_1 \cap L_2$) indistinguishable from the tuple $(\mathsf{hp}_1, \mathsf{hp}_2, g)$.

B.7.3 Preservation of the Uniformity and Independence Properties.

If the two underlying smooth hash systems verify 1-uniformity (resp. 2-uniformity), then the smooth hash system for conjunction or disjunction verifies these properties. If the two underlying smooth hash systems verify 1-independence (resp. 2-independence), then the smooth hash system for conjunction verifies these properties. We insist on the fact that independence does not propagate to disjunction, since the hash value (that needs both aux and x) is included in the projected key.

Proof: We only prove the result for 1-uniformity for disjunction (the proof is the same in the other cases —excepted independence for disjunction, where the result does not hold). Then, if (ρ, aux) are the parameters of the languages and x and x' belong to L, $D_{1,\rho,aux,x} \approx D_{1,\rho,aux,x'}$ and $D_{2,\rho,aux,x} \approx D_{2,\rho,aux,x'}$. Due to the form of hp, this ensures that the first two element of hp are indistinguishable. We now have to consider the third part. Without loss of generality, we can suppose that $x \in L_1$. Then, due to the pseudo-randomness of the first smooth-hash, the value $\mathsf{Hash}_1(\mathsf{hk}_1; \rho, aux, x)$ is computationally indistinguishable from uniform. This is at least the same for the value $\mathsf{Hash}_2(\mathsf{hk}_2; \rho, aux, x)$. Since the projected keys depend on independent random values and languages, both parts of the $\oplus$ are independent: the value $\mathsf{Hash}_L(\mathsf{hk}_L; \rho, aux, x)$ is then indistinguishable from uniform. As a result, $D_{\rho,aux,x} \approx D_{\rho,aux,x'}$. ▮

B.8 Appendix: Proofs for Commitment in Section B.5

Extractability. The extraction key is sk, the ElGamal decryption key (or the Cramer-Shoup one in the non-malleable version of the primitive, also used for the ElGamal decryption). First, the simulator tries to decrypt all the $b_{i,\delta}[j]$ into $x_{i,\delta}[j]$ and aborts if one of them is not either 0 or 1. It then builds up the $x_{i,\delta}$, and checks whether $a_i = g^{x_{i,0}}$ or $a_i = g^{x_{i,1}} y_i$, which makes it recover π_i (unless none or both are satisfied). This extraction only fails in three cases:

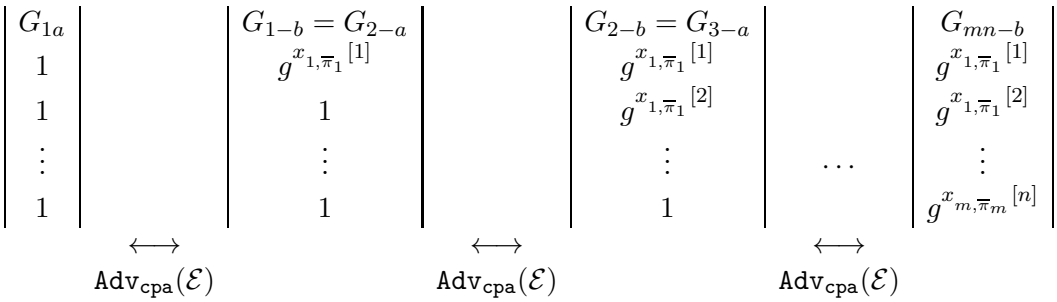
- First, if the ciphertexts do not encrypt 0 or 1;
- Second, if a_i satisfies none of the equalities;
- Third, if a_i satisfies both equalities.

The two first reasons will be excluded in the language L, while the third one would break the binding property of the Pedersen commitment, which leads to the discrete logarithm of some y_i in base g. It thus happens only with negligible probability.

Equivocability. Note that, with the knowledge of the discrete logarithms of $(y_1, \ldots, y_m)$, one is able to compute, for all $i \in \{1, \ldots, m\}$, both $x_{i,0}$ and $x_{i,1}$. The equivocation thus consists in

computing, for all i, an encryption of both $x_{i,0}$ and $x_{i,1}$ (and not that of 0). Provided one does not erase any random, this allows one to change its mind and open each commitment on any π_i (0 or 1).

We now prove that committing to all the bit-string π in a unique commitment does not change the view of an adversary. The proof is based on a Left-or-Right hybrid argument, see [BDJR97]. We suppose the existence of an oracle answering with either $\mathcal{E}(g^x)$ or $\mathcal{E}(g^0)$ when provided with g^x (and g^0 is implicitly given). We then define hybrid games in which the encryptions for $x_{i,\pi_i}[j]$ in the commitment are computed with the help of this oracle. The first hybrid game, in which the oracle always encrypts g^0, is equivalent to the real computation, where only one bit-string is committed (perfectly binding). Similarly, the last one, in which it always encrypts g^x, is equivalent to the simulation, where all the bit-strings are committed (equivocable).



Hiding. This property simply follows from the equivocability of the commitment.

Binding. First suppose that we do not know the discrete logarithms of $(y_1, \ldots, y_m)$ in base g and that the adversary has managed to send a commitment that can be opened on π and on π', then, for some bit $i = 1, \ldots, m$, $\pi_i \neq \pi'_i$, and the adversary is able to give us $x_i \neq x'_i$ such that $g^{x_i} y^{\pi_i} = g^{x'_i} y^{\pi'_i}$. This event thus boils down to breaking the discrete logarithm problem, which happens only with negligible probability. This means that if equivocability is not used, the commitment is (computationally) binding, under the discrete logarithm problem.

Now, since a real commitment (with a zero encryption) and an equivocable commitment (with no zero encryption) are indistinguishable for an adversary, the view of equivocable commitments does not help the adversary to break the binding property, or otherwise it would break the IND-CPA property of the underlying encryption scheme.

B.9 Appendix: A Non-Malleable Conditionally-Extractable Equivocable Commitment

In this section, we show how to enhance the previous commitment schemes as described in Section B.5 with non-malleability: briefly, one simply needs to extend the ElGamal commitment to the labeled Cramer-Shoup one together with one-time signatures. As before, if b is a bit, we denote its complement by $\overline{b}$ (i.e., $\overline{b} = 1 - b$). We furthermore denote by $x[i]$ the i^{th} bit of the bit-string x.

B.9.1 Non-Malleability

Non-malleability is a usual requirement for encryption schemes or commitments [GL03]. We thus now aim at achieving this property. We thus use labeled Cramer-Shoup encryption instead of ElGamal, and we add a one-time signature. Using the results of Dodis and Katz [DK05] for chosen-ciphertext security of multiple encryption, one can easily show that the chosen-ciphertext

security (and thus non-malleability) of the combined encryption scheme used to compute a ciphertext vector $(b_1, \ldots, b_m)$ follows trivially from the chosen-ciphertext security of the underlying labeled Cramer-Shoup scheme and strong unforgeability of the one-time signature scheme used to link all the ciphertexts together.

More precisely, if M is defined as before, and ℓ is a label, the commitment $\mathsf{comCS}^{\ell}_{\mathsf{pk}}(M)$ is obtained as follows. First, the user generates a key pair $(\mathrm{VK}, \mathrm{SK})$ for a one-time signature scheme. Then, it computes the following values, with $\ell' = \ell \circ \mathrm{VK}$:

$$\forall i \qquad b_i = \mathsf{CS}^{+\,\ell'}_{\mathsf{pk}}(M_i \cdot 2^{i-1}, r_i) = (u_{1,i} = g_1^{r_i}, u_{2,i} = g_2^{r_i}, e_i = h^{r_i} g^{M_i \cdot 2^{i-1}}, v_i = (cd^{\theta_i})^{r_i}).$$

Defining $\mathbf{b} = (b_1, \ldots, b_m)$, it computes $\sigma = \mathsf{Sign}(\mathrm{SK}, \mathbf{b})$. The final commitment is then defined as $\mathsf{comCS}^{\ell'}_{\mathsf{pk}}(M) = (\mathbf{b}, \mathrm{VK}, \sigma)$. One can obtain, from any b_i, an ElGamal encryption of $M_i \cdot 2^{i-1}$: $B_i = \mathsf{EG}^{+}_{\mathsf{pk}}(M_i \cdot 2^{i-1}, r_i) = (u_{1,i} = g_1^{r_i}, e_i = h^{r_i} g^{M_i \cdot 2^{i-1}})$. The homomorphic property of the encryption scheme allows to obtain $B = \mathsf{EG}^{\times}_{\mathsf{pk}}(g^M, \sum r_i) = \mathsf{EG}^{+}_{\mathsf{pk}}(M, \sum r_i) = (\prod u_{1,i}, \prod e_i) = \prod B_i$.

In order to define the smooth projective hashing associated with this commitment scheme, we recall the family of smooth projective hashing functions for the underlying labeled Cramer-Shoup encryption scheme, as defined in [GL03].

Let $X' = G^4$ and $L' = L_{(\mathbf{CS}^+, \rho), (\ell, M)}$ be the language of the elements C such that C is a valid Cramer-Shoup encryption of M under the label ℓ (aux is defined as (ℓ, M)). Under the DDH assumption, this is a hard subset membership problem. Denoting by $C = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(M, r) = (u_1, u_2, e, v)$, the associated smooth hash system is the following:

$$\begin{aligned}
\mathsf{HashKG}((\mathbf{CS}^+, \rho), (\ell, M)) &= \mathsf{hk} = (\gamma_1 \grave{\mathrm{u}}, \gamma_2 \grave{\mathrm{u}}, \gamma_3 \grave{\mathrm{u}}, \gamma_4 \grave{\mathrm{u}}) \xleftarrow{\$} \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q \\
\mathsf{ProjKG}(\mathsf{hk}; (\mathbf{CS}^+, \rho), (\ell, M), C) &= \mathsf{hp} = (g_1)^{\gamma_1 \grave{\mathrm{u}}} (g_2)^{\gamma_2 \grave{\mathrm{u}}} (h)^{\gamma_3 \grave{\mathrm{u}}} (cd^{\theta})^{\gamma_4 \grave{\mathrm{u}}} \\
\mathsf{Hash}(\mathsf{hk}; (\mathbf{CS}^+, \rho), (\ell, M), C) &= (u_1)^{\gamma_1 \grave{\mathrm{u}}} (u_2)^{\gamma_2 \grave{\mathrm{u}}} (eg^M)^{\gamma_3 \grave{\mathrm{u}}} (v)^{\gamma_4 \grave{\mathrm{u}}} \\
\mathsf{ProjHash}(\mathsf{hp}; (\mathbf{CS}^+, \rho), (\ell, M), C; r) &= (\mathsf{hp})^r
\end{aligned}$$

From these definitions, we consider the language: $L_{(\mathbf{CS}^+, \rho), (\ell, 0 \vee 1)} = L_{(\mathbf{CS}^+, \rho), (\ell, 0)} \cup L_{(\mathbf{CS}^+, \rho), (\ell, 1)}$, where $L_{(\mathbf{CS}^+, \rho), (\ell, 0)} = \{C \mid \exists r \; C = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(0, r)\}$ and $L_{(\mathbf{CS}^+, \rho), (\ell, 1)} = \{C \mid \exists r \; C = \mathsf{CS}^{+\,\ell}_{\mathsf{pk}}(1, r)\}$, and we want to define a smooth hash system showing that

$$\forall i \quad b_i \in L_{(\mathbf{CS}^+, \rho), (\ell', 0 \vee 1)} \qquad \text{and} \qquad B = \prod B_i \in L_{(\mathbf{EG}^{\times}, \rho), g^M}.$$

Note that, for all i, the first membership also implies that $B_i \in L_{(\mathbf{EG}^+, \rho), 0 \vee 1}$, since the ElGamal ciphertexts B_i are extracted from the corresponding Cramer-Shoup ciphertexts b_i: It is thus enough to check the validity of the latter membership to get extractability, which means that one can extract the committed private key M, associated to the public key g^M. The signature has of course to be verified too, but this can be publicly performed, from VK.

B.9.2 Equivocability

We here describe our commitment for completeness, but note that it is very similar to the one described in Section B.5.

Description of the Commitment. Our scheme uses the labeled Cramer-Shoup public-key encryption scheme and a one-time signature scheme, to achieve non-malleability as explained in the previous section. More precisely, the specific example given in this section relies on the labeled version of the Cramer-Shoup encryption scheme [CS98], used for each bit of the bit-string, seen as an extension of the homomorphic ElGamal encryption [ElG85b].

Let the input of the committing algorithm be a bit-string $\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$ and a label ℓ. The algorithm works as follows:

- For $i = 1, \ldots, m$, it chooses a random value $x_{i,\pi_i} = \sum_{j=1}^{n} x_{i,\pi_i}[j] \cdot 2^{j-1}$ and sets $x_{i,\overline{\pi}_i} = 0$. It also generates a key pair $(\mathrm{VK}, \mathrm{SK})$ for a one-time signature scheme.

- For $i = 1, \ldots, m$, it commits to π_i, using the random x_{i,π_i}: $a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}) = g^{x_{i,\pi_i}} y_i^{\pi_i}$ and defining $\mathbf{a} = (a_1, \ldots, a_m)$.

- For $i = 1, \ldots, m$, using $\ell_i = \ell \circ \mathrm{VK} \circ \mathbf{a} \circ i$, it computes the Cramer-Shoup commitments (see the previous section) of $x_{i,\delta}$, for $\delta = 0, 1$:

  $$(\mathbf{b}_{i,\delta} = (b_{i,\delta}[j])_j, \mathrm{VK}, \sigma_{i,\delta}) = \mathsf{comCS}^{\ell_i}_{\mathsf{pk}}(x_{i,\delta}), \text{ where } b_{i,\delta}[j] = \mathsf{CS}^{+\,\ell_i}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]).$$

  The computation of the $b_{i,\delta}[j]$ implicitly defines $B_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j])$, from which one can directly extract an encryption $B_{i,\delta}$ of $x_{i,\delta}$: $B_{i,\delta} = \prod_j B_{i,\delta}[j] = \mathsf{EG}^{+}_{\mathsf{pk}}(x_{i,\delta}, r_{i,\delta})$, where $r_{i,\delta}$ is the sum of the random coins $r_{i,\delta}[j]$.

The entire random string for this commitment is (where n is the bit-length of the prime order q of the group G) $R = (x_{1,\pi_1}, (r_{1,0}[1], r_{1,1}[1], \ldots, r_{1,0}[n], r_{1,1}[n]), \ldots, x_{m,\pi_m}, (r_{m,0}[1], \ldots, r_{m,1}[n]), \mathrm{SK})$. From which, all the values $r_{i,\overline{\pi}_i}[j]$ can be erased, letting the opening data (witness of the committed value) become $\mathsf{w} = (x_{1,\pi_1}, (r_{1,\pi_1}[1], \ldots, r_{1,\pi_1}[n]), \ldots, x_{m,\pi_m}, (r_{m,\pi_m}[1], \ldots, r_{m,\pi_m}[n]))$.

The output of the committing algorithm, of the bit-string π, with the label ℓ, and using the random R, is $\mathsf{com}_\rho(\ell, \pi; R) = (\ell, \mathbf{a}, \mathbf{b}, \mathrm{VK}, \sigma)$, where $\mathbf{a} = (a_i = \mathsf{comPed}(\pi_i, x_{i,\pi_i}))_i$, $\mathbf{b} = (b_{i,\delta}[j] = \mathsf{CS}^{+\,\ell_i}_{\mathsf{pk}}(x_{i,\delta}[j] \cdot 2^{j-1}, r_{i,\delta}[j]))_{i,\delta,j}$, and $\sigma = (\sigma_{i,\delta} = \mathsf{Sign}(\mathrm{SK}, (b_{i,\delta}[j])_j))_{i,\delta}$.

Properties. This commitment is opened as in Section B.5 and the proofs given in Section B.8 still hold.

Furthermore, let us show now that the view of equivocable commitments does not help the adversary to build a valid but non-extractable commitment (it could be open in any way, and thus extraction leads to many possibilities) due to the CCA property of the encryption. As in Appendix B.8, we use a Left-or-Right argument, see [BDJR97]. The "left" oracle, which always provides the player with an encryption $\mathcal{E}(g^0)$, is equivalent to the game where no equivocable commitments are available, and the "right" oracle, which always provides him with $\mathcal{E}(g^x)$, is equivalent to the game where the commitments are equivocable. Then, if the adversary was more likely to build a valid but non-extractable commitment in the latter case than in the former one, one could construct a distinguisher to the Left-or-Right oracles. However, contrarily to the proof in Appendix B.8, a decryption oracle is required to check whether the commitment is valid but non-extractable (whereas in Appendix B.8 the adversary breaks the binding property with two different opening values).

As a consequence, producing valid but non-extractable commitments is not easier when equivocable commitments are provided to the adversary, than when no equivocable commitments are provided, under the IND-CCA security of the encryption scheme. Furthermore, a valid but non-extractable commitment, with a decryption oracle leads to two different opening values for the commitment, and thus to an attack against the binding property, which relies on the discrete logarithm problem.

The additional property is non-malleability, and it is guaranteed by the labeled Cramer-Shoup encryption scheme (IND-CCA) and the one-time signature, as already explained [DK05]. More precisely, for the results of Dodis and Katz [DK05] for chosen-ciphertext security of multiple encryption, one can easily show that the chosen-ciphertext security of the combined encryption scheme used to compute ciphertext vector $\mathbf{b}$ follows trivially from the chosen-ciphertext security of the underlying labeled Cramer-Shoup scheme and strong unforgeability of the one-time signature scheme used to link all the ciphertexts together.

B.9.3 The Associated Smooth Projective Hash Function

As noticed above, our new commitment scheme is conditionally extractable (one can recover the $x_{i,\delta}$'s, and then π), under the conditions that all the Cramer-Shoup ciphertexts encrypt either 0 or 1, and the a_i is a commitment of either 0 or 1, with random $x_{i,0}$ or $x_{i,1}$.

Since we want to apply it later to the password-based setting, we want to make the two hash values (direct computation and the one from the projected key) to be the same if the two parties use the same password π, but perfectly independent if they use different passwords: using their password $\pi = \sum_{i=1}^{m} \pi_i \cdot 2^{i-1}$, one furthermore wants to ensure that each a_i is actually a Pedersen commitment of π_i using the encrypted random x_{i,π_i}, and thus $g^{x_{i,\pi_i}} = a_i/y_i^{\pi_i}$: the extracted x_{i,π_i} is really the private key M related to a given public key g^M that is $a_i/y_i^{\pi_i}$ in our case. Using the same notations as in Section B.9.1, we want to define a smooth hash system showing that, for all i, δ, j, $b_{i,\delta}[j] \in L_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}$ and, for all i, $B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),(a_i/y_i{}^{\pi_i})}$, where $B_{i,\pi_i} = \prod_j B_{i,\pi_i}[j]$. As before, note that, for all i, δ, j, the first membership also implies that $B_{i,\delta}[j] \in L_{(\mathbf{EG}^+,\rho),0\vee1}$ since this value is extracted from the corresponding value $b_{i,\delta}[j]$: It is thus enough to check the validity of the latter.

Combinations of these smooth hashes. Let C be the above commitment of π using label ℓ and randomness R as defined in Section B.9.2. We now precise the language $L_{\rho,(\ell,\pi)}$, consisting informally of all the valid commitments "of good shape":

$$L_{\rho,(\ell,\pi)} = \left\{ C \;\middle|\; \begin{array}{l} \exists R \text{ such that } C = \mathsf{com}_\rho(\ell, \pi, R) \\ \quad \text{and } \forall i\ \forall \delta\ \forall j \quad b_{i,\delta}[j] \in L_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)} \\ \quad \text{and } \forall i \quad B_{i,\pi_i} \in L_{(\mathbf{EG}^\times,\rho),a_i/y_i{}^{\pi_i}} \end{array} \right\}$$

The smooth hash system for this language relies on the smooth hash systems described in the previous subsection, using the generic construction for conjunctions and disjunctions as described in Section B.3. More precisely, adding the values $B_{i,\delta}$ easily computed from the values $b_{i,\delta}[j]$, the commitment can be converted into a tuple of the following form (we omit the signature part, since it has to be verified before applying any hashing computation):

$$\begin{array}{ll} (\ell, & a_1, \ldots, a_m, \quad b_{1,0}[1], b_{1,1}[1], \ldots, b_{1,0}[n], b_{1,1}[n], \ldots, b_{m,0}[1], b_{m,1}[1], \ldots, b_{m,0}[n], b_{m,1}[n], \\ & \qquad B_{1,0}, B_{1,1}, \ldots, B_{m,0}, B_{m,1}) \in \{0,1\}^* \times G^m \times (G^4)^{2mn} \times (G^2)^{2m}. \end{array}$$

We denote by $L^{i,\delta,j}_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}$ the language that restricts the value $b_{i,\delta}[j]$ to be a valid Cramer-Shoup encryption of either 0 or 1:

$$\begin{array}{l} \underbrace{\{0,1\}^*}_{\ell} \times \underbrace{G^m}_{\mathbf{a}} \times \underbrace{(G^4)^{2n}}_{b_{1,*}[*]} \times \ldots \times (\underbrace{G^4}_{b_{i,0}[1]} \times \underbrace{G^4}_{b_{i,1}[1]} \times \ldots \times \underbrace{L_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}}_{b_{i,\delta}[j]} \times \ldots \times \underbrace{G^4}_{b_{i,0}[n]} \times \underbrace{G^4}_{b_{i,1}[n]}) \\ \qquad \times \ldots \times \underbrace{(G^4)^{2n}}_{b_{1,m}[*]} \times \underbrace{(G^2)^{2m}}_{B_{*,*}} \end{array}$$

and by $L^i_{(\mathbf{EG}^\times,\rho)}$ the language that restricts the B_{i,π_i} ElGamal ciphertexts:

$$\text{if } \pi_i = 0, L^i_{(\mathbf{EG}^\times,\rho)} = \underbrace{\{0,1\}^*}_{\ell} \times \underbrace{G^m}_{\mathbf{a}} \times \underbrace{(G^4)^{2mn}}_{b_{*,*}[*]} \times \underbrace{(G^2)^2}_{B_{1,*}} \times \ldots \times (\underbrace{L_{(\mathbf{EG}^\times,\rho),a_i}}_{B_{i,0}} \times \underbrace{G^2}_{B_{i,1}}) \times \ldots \times \underbrace{(G^2)^2}_{B_{m,*}}$$

$$\text{if } \pi_i = 1, L^i_{(\mathbf{EG}^\times,\rho)} = \underbrace{\{0,1\}^*}_{\ell} \times \underbrace{G^m}_{\mathbf{a}} \times \underbrace{(G^4)^{2mn}}_{b_{*,*}[*]} \times \underbrace{(G^2)^2}_{B_{1,*}} \times \ldots \times (\underbrace{G^2}_{B_{i,0}} \times \underbrace{L_{(\mathbf{EG}^\times,\rho),a_i/y_i}}_{B_{i,1}}) \times \ldots \times \underbrace{(G^2)^2}_{B_{m,*}}.$$

Then, our language $L_{\rho,(\ell,\pi)}$ is the conjunction of all these languages, where the $L^{i,\delta,j}_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}$'s are disjunctions:

$$L_{\rho,(\ell,\pi)} = \left(\bigcap_{i,\delta,j} L^{i,\delta,j}_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}\right) \cap \left(\bigcap_i L^i_{(\mathbf{EG}^\times,\rho)}\right).$$

Properties: Uniformity and Independence. With such a commitment and smooth hash function, it is possible to improve the establishment of a secure channel between two players, from the one presented Section B.4.3. More precisely, two parties can agree on a common key if they both share a common (low entropy) password π. However, a more involved protocol than the one proposed in Section B.4.3 is needed to achieve all the required properties of a password-authenticated key exchange protocol, as it will be explained and proven in the next appendix.

Nevertheless, there may seem to be a leakage of information because of the language that depends on the password π: the projected key hp seems to contain some information about π, that can be used in another execution by an adversary. Hence the independence and uniformity notions presented Section B.3.4, which ensure that hp does not contain any information about π:

- for the languages $L^i_{(\mathbf{EG}^\times,\rho)}$, the smooth hash functions satisfy the *2-independence* property, since the projected key for a language $L_{(\mathbf{EG}^\times,\rho),M}$ depends on the public key (and thus ρ) only. One thus generates one key for each pair $(B_{i,0}, B_{i,1})$ only, and uses the correct ciphertext according to actual/wanted π_i when evaluating the hash value. But this distinction on π_i appears in the computation of the hash value only, that is pseudo-random. The projected key is totally independent of π_i.
- for the languages $L^{i,\delta,j}_{(\mathbf{CS}^+,\rho),(\ell_i,0\vee1)}$, the smooth hash functions satisfy the *2-uniformity* property only, but not *2-independence*. Therefore, the projected key and (aux, ρ) are statistically independent, but the former depends, in its computation, of the latter. If we want the equivocability property for the commitment (as needed in the next section), we have to include all the pairs $(b_{i,0}[j], b_{i,1}[j])$, and not only the $b_{i,\pi_i}[j]$, so that we can open later in any way. Because of 2-uniformity instead of 2-independence, we need a key for each element of the pair, and not only one as above.

Estimation of the Complexity. Globally, each operation (commitment, projected key, hashing and projected hashing) requires $\mathcal{O}(mn)$ exponentiations in G, with small constants (at most 16).

Let us first consider the commitment operation, on a m-bit secret π, over a n-bit group G. One has first to make m Pedersen commitments of one bit (m exponentiations) and then $2mn$ additive Cramer-Shoup encryptions (2 exponentiations and 2 multi-exponentiations each, and thus approximately the cost of $8mn$ exponentiations).

About the smooth hash function, the hash key generation just consists in generating random elements in $\mathbb{Z}_q$: 8 for each Cramer-Shoup ciphertext, in order to show that they encrypt either 0 or 1, and 2 for each pair of ElGamal ciphertexts, then globally $2m(8n+1)$ random elements in $\mathbb{Z}_q$. The projected keys need exponentiations: $m(4n+1)$ multi-exponentiations, and $4mn$ hash evaluations (one multi-exponentiation each). Then, globally, the cost of $m(8n+1)$ exponentiations in G is required for the projected key. Finally, the hash computations are essentially the same using the hash key or the projected key, since for the sub-functions, the former consists of one multi-exponentiation and the latter consists of 1 exponentiation. They both cost $m(4n+1)$ exponentiations, after the multiplications needed to compute the B_{i,π_i}, which are negligible.

B.10 Appendix: A New Adaptively-Secure PAKE Protocol in the UC Framework

B.10.1 Password-Based Key Exchange and Universal Composability

The Password-Based Key Exchange Functionality.

In this section, we present the password-based key-exchange functionality $\mathcal{F}_{pwKE}$ (see Figure B.1) first described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted, then they both end up with the same uniformly-distributed session key, and the adversary learns nothing about it (except that it was indeed generated). However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), then it is granted the right to fully determine its session key. Note that as soon as a party is corrupted, the adversary learns its key: There is in fact nothing lost by allowing it to determine the key.

In addition, the players become aware of a failed attempt of the adversary at guessing a password. This is modeled by marking the session as `interrupted`. In this case, the two players are given independently-chosen random keys.

A session that is nor `compromised` nor `interrupted` is called `fresh`. In such a case, the two parties receive the same, uniformly distributed session key.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password is used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

The KOY/GL Protocol.

The starting point of our protocol is the password-based key exchange protocol of Katz, Ostrovsky and Yung [KOY01], generalized by Gennaro and Lindell in [GL03]. At a high level, the players in the KOY/GL protocol exchange CCA-secure encryptions of the password, under the public-key found in the common reference string, which is essentially a commitment of the password. Then, they compute the session key by combining smooth projective hashes of the two password/ciphertext pairs. More precisely, each player chooses a hashing key for a smooth projective hash function and sends the corresponding projected key to the other player. Each player can thus compute the output of its own hash function with the help of the hashing key, and the output of the other one using the projected key and its knowledge of the randomness that was used to generate the ciphertext of the password. All the flows generated by a party are linked together with a one-time signature, generated in the last flow, but which public key is included in the label of the CCA-secure encryption of the password.

To understand informally why this protocol is secure, first consider the case in which the adversary plays a passive role. In this case, the pseudo-randomness property of the smooth hash function ensures that the value of the session key will be computationally indistinguishable from uniform since the adversary does not know the randomness that was used to encrypt the password. Now imagine the case in which the adversary provides the user with an encryption of the wrong password. In this case, the security of the protocol will rely on the smoothness of the hash functions, which ensures that the session key will be random and independent of

The functionality $\mathcal{F}_{pwKE}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving a query (`NewSession`, $sid, P_i, P_j, pw, role$) from party P_i:**

  Send (`NewSession`, $sid, P_i, P_j, role$) to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record (P_j, P_i, pw'), then record (P_i, P_j, pw) and mark this record `fresh`.

- **Upon receiving a query (`TestPwd`, sid, P_i, pw') from the adversary S:**

  If there is a record of the form (P_i, P_j, pw) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If $pw \neq pw'$, mark the record `interrupted` and reply with "wrong guess".

- **Upon receiving a query (`NewKey`, sid, P_i, sk) from the adversary S:**

  If there is a record of the form (P_i, P_j, pw), and this is the first `NewKey` query for P_i, then:

  - If this record is `compromised`, or either P_i or P_j is corrupted, then output (sid, sk) to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw') with $pw' = pw$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output (sid, sk') to P_i.
  - In any other case, pick a new random key sk' of length k and send (sid, sk') to P_i.

  Either way, mark the record (P_i, P_j, pw) as `completed`.

Figure B.1: The password-based key-exchange functionality $\mathcal{F}_{pwKE}$

all former communication. Thus, in order to be successful, the adversary has to generate the encryption of the correct password. To do so, the adversary could try to copy or modify existing ciphertexts. Since the encryption scheme is CCA-secure, and thus non-malleable, modifying is not really a possibility. Copying does not help either since either the label used for encryption will not match (making the session key look random due to the smoothness property) or the signature will be invalid (in the case where the adversary changes the projection keys without changing the label and hence the verification key). As a result, the only successful strategy left for the adversary is essentially to guess the password and perform the trivial online dictionary attack, as desired.

Extending the protocol to the UC Framework (Static Case). As noted by Canetti *et al* in [CHK+05], the KOY/GL protocol is not known to achieve UC security: the main issue is that the ideal-model simulator must be able to extract the password used by the adversary. One could think that, since the simulator has control over the common reference string, it knows all private keys corresponding to the public keys and can thus decrypt all ciphertexts sent by the adversary and recover its password.

But indeed, this doesn't not seem to be sufficient. In the case where the adversary begins to play (*i.e.* it impersonates the client), everything works well: The simulator decrypts the ciphertext generated by the adversary and can thus recover the password it has used. If the guess of the adversary is incorrect (that is, the password is the wrong one), then the smoothness

of the hash functions leads to random independent session keys. Otherwise, if the guess is correct, the execution can continue as an honest one would do (the simulator has learned which password to use).

However, let's now suppose that the simulator has to start the game, on behalf of the client. Here, the simulator needs to send an encryption of the password before having seen anything coming from the adversary. As described above, it recovers the password used by the adversary as soon as the latter has sent its value, but this is too late. If it turns out that the guess of the adversary is incorrect, there is no problem thanks to the smoothness, but otherwise, the simulator is stuck with an incorrect ciphertext and will not be able to predict the value of the session key.

To overcome this problem, the authors of [CHK+05] made the client send a pre-flow which also contains an encryption of the password. The server then sends its own encryption, and finally the client sends another encryption (this time the simulator is able to use the correct password, recovered from the value sent by the adversary), as well as a zero-knowledge proof claiming that both ciphertexts are consistent and encrypt the same password. The first flow is never used in the remaining of the protocol. This solves the problem since on the one hand, the simulator is of course able to give a valid proof of a false statement, and on the other hand, the first flow will never be used afterwards.

B.10.2 Description of the Protocol

As explained above, Canetti *et al.* [CHK+05] proposed a simple variant of the Gennaro-Lindell methodology [GL03] that is provably secure in the UC framework. Though quite efficient, their protocol is not known to be secure against adaptive adversaries. The only one PAKE adaptively-secure in the UC framework was proposed by Barak *et al.* [BCL+05] using general techniques from multi-party computation. It thus leads to quite inefficient schemes.

In the following, we use our non-malleable conditionally-extractable and equivocable commitment scheme with an associated smooth projective hash function family, in order to build an efficient PAKE scheme, adaptively-secure in the common reference string model under standard complexity assumptions, in the UC framework.

The complete description of the protocol can be found in Figure B.3, whereas an informal sketch is presented in Figure B.2. We use the index I for a value related to the client Alice, and J for a value related to the server Bob.

Correctness. In an honest execution of the protocol, if the players share the same password ($\mathsf{pw}_\mathrm{I} = \mathsf{pw}_\mathrm{J} = \mathsf{pw}$), it is easy to verify that both players will terminate by accepting and computing the same values for the session key, equal to $\mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \mathsf{pw}, \ell_\mathrm{J}, \mathsf{com}_\mathrm{J}) + \mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \mathsf{pw}, \ell_\mathrm{I}, \mathsf{com}_\mathrm{I})$.

Security. The intuition behind the security of our protocol is quite simple and builds on that of Gennaro-Lindell protocol. The key point in order to achieve adaptive security is the use of the commitment, which allows for extraction and equivocation at any moment, thus not requiring the simulator to be aware of future corruptions. The following theorem, which full proof will be given in Appendix B.10.3, states that the protocol is UC-secure. The ideal functionality $\mathcal{F}_{pwKE}$ has been presented in Figure B.1 and described in Appendix B.10.1. Since we use the joint state version of the UC theorem, we implicitly consider the multi-session extension of this functionality. In particular, note that the passwords of the players depend on the session considered. For sake of simplicity, we denote them by pw_I and pw_J, but one should implicitly understand $\mathsf{pw}_{\mathrm{I,ssid}}$ and $\mathsf{pw}_{\mathrm{J,ssid}}$.

Theorem B.10.1 Let com be the non-malleable (conditionally) extractable and equivocable committing scheme described in Section B.9, $\mathcal{H}$ be a family of smooth hash functions with respect

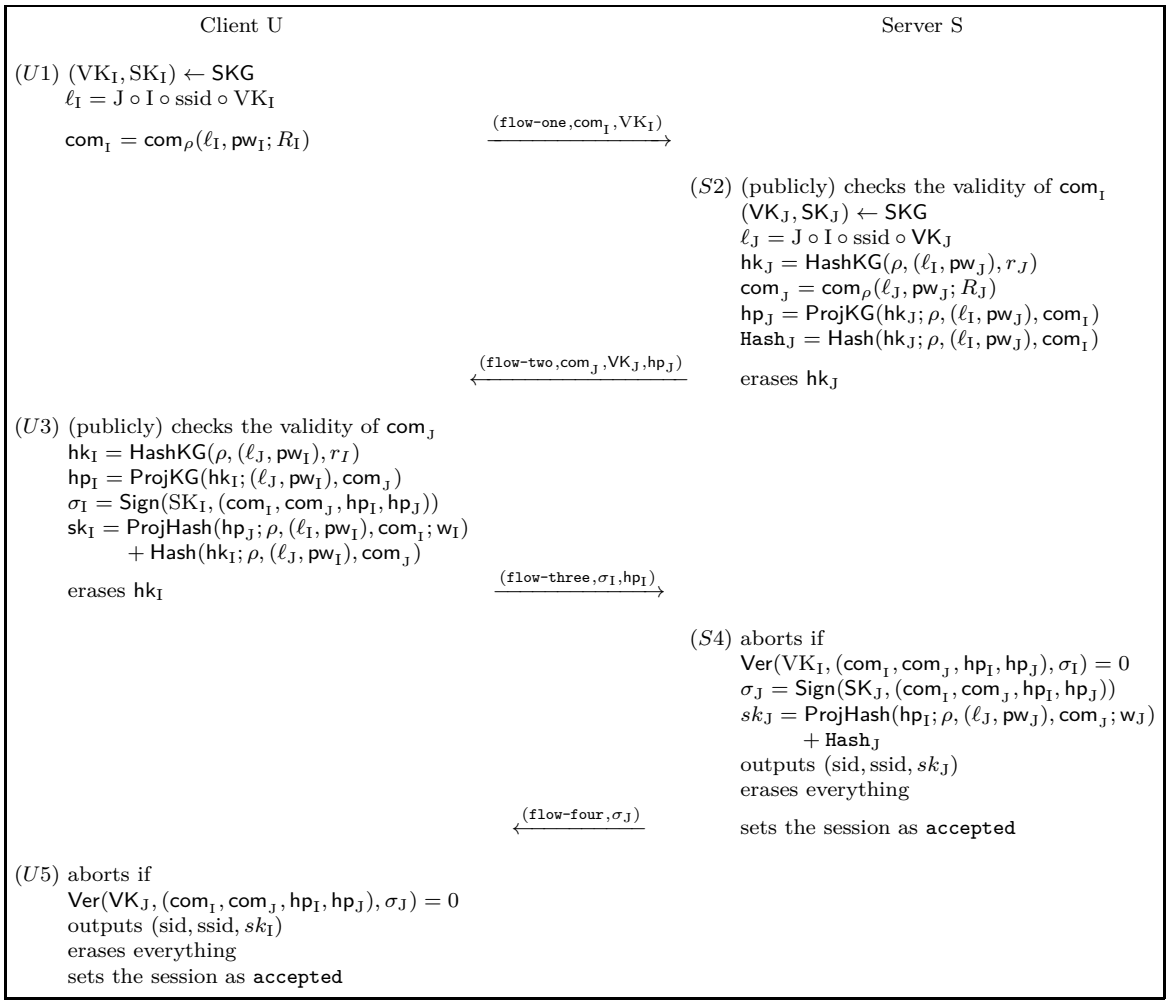


Figure B.2: Description of the protocol for players $(P_\mathrm{I}, \mathrm{ssid})$, with index I and password pw_I and $(P_\mathrm{J}, \mathrm{ssid})$, with index J and password pw_J. At the end of rounds 1 and 2, the players will erase the part of the random values R_I and R_J used in the commitment which is not needed in the following rounds, keeping only $\mathsf{w_I}$ and $\mathsf{w_J}$.

to this commitment, and $\mathcal{SIG}$ be a one-time signature scheme. Denote by $\widehat{\mathcal{F}}_{pwKE}$ the multi-session extension of the functionality $\mathcal{F}_{pwKE}$ of password-based key-exchange, and let $\mathcal{F}_{CRS}$ be the ideal functionality that provides a common reference string $(G, \mathsf{pk}, (y_1, \ldots, y_m), \mathsf{Extract})$ to all parties, where G is a cyclic group, $y_1, \ldots, y_m$ random elements from this group, pk a public key for the Cramer-Shoup scheme and $\mathsf{Extract}$ a randomness extractor. Then, the above protocol securely realizes $\widehat{\mathcal{F}}_{pwKE}$ in the $\mathcal{F}_{CRS}$-hybrid model, in the presence of adaptive adversaries.

B.10.3 Proof of Theorem B.10.1

Sketch of Proof. In order to prove Theorem B.10.1, we need to construct, for any real-world adversary $\mathcal{A}$ (interacting with real parties running the protocol), an ideal-world adversary $\mathcal{S}$ (interacting with dummy parties and the functionality $\mathcal{F}_{pwKE}$) such that no environment $\mathcal{Z}$ can distinguish between an execution with $\mathcal{A}$ in the real world and $\mathcal{S}$ in the ideal world with non-negligible probability.

We first describe two hybrid queries that are going to be used in the games. The `GoodPwd` query checks whether the password of some player is the one we have in mind or not. The `SamePwd` query checks if the players share the same password, without disclosing it. In some games the simulator has actually access to the players. In such a case, a `GoodPwd` (or a `SamePwd`) query can be easily implemented by letting the simulator look at the passwords owned by the oracles. When the players are entirely simulated, $\mathcal{S}$ will replace the queries above with `TestPwd` and `NewKey` queries.

We say that a flow is *oracle-generated* if it was sent by an honest player and arrives without any alteration to the player it was meant to. We say it is *non-oracle-generated* otherwise, that is either if it was sent by an honest player and modified by the adversary, or if it was sent by a corrupted player or a player impersonated by the adversary (more generally denoted by *attacked* player, that is, a player whose password is known to the adversary).

We incrementally define a sequence of games starting from the one describing a real execution of the protocol and ending up with game $\boldsymbol{G}_8$ which we prove to be indistinguishable with respect to the ideal experiment. For the sake of clarity, we will use the following notation: the client is Alice (hence She), the server is Bob (hence He), and we use It for the adversary $\mathcal{A}$ and the simulator $\mathcal{S}$.

- $\boldsymbol{G}_0$ is the real game.
- From $\boldsymbol{G}_1$, $\mathcal{S}$ is allowed to program the CRS.
- From $\boldsymbol{G}_2$, $\mathcal{S}$ always extracts the password committed to by the adversary and aborts when the extraction fails due to the commitment being valid for two or more passwords. This would lead to an attack against the binding property of the commitment scheme.
- From $\boldsymbol{G}_3$, $\mathcal{S}$ simulates all the commitments and makes them equivocable granted the simulated CRS. The commitment remains binding and hiding for the environment and the adversary, which follows from the CCA2-property of the encryption. As a side note, the knowledge of the passwords is not necessary anymore for the simulation of the committing step.
- From $\boldsymbol{G}_4$, $\mathcal{S}$ simulates the honest client without using her password anymore (except for the hybrid queries). She is given a random key in $(U3)$ in the case where `flow-two` was oracle-generated and the server was not corrupted. If the server was corrupted, $\mathcal{S}$ recovers his password pw_J and makes a call to the `GoodPwd` functionality for Alice. If it is incorrect, Alice is also given a random key, but if it is correct, the key is computed honestly using that password. If no password is recovered, Alice is also given a random key. Alice then aborts in $(U5)$ if the signature of the server is invalid. If the server was corrupted before $(U5)$, $\mathcal{S}$ recovers his password and does exactly the same as previously described. This is indistinguishable from the former game due to the pseudo-randomness of the hash function.
- From $\boldsymbol{G}_5$, in the case where `flow-two` was not oracle-generated, $\mathcal{S}$ extracts pw_J from com_J and proceeds as described in $\boldsymbol{G}_4$: it asks a `GoodPwd` query for Alice and provides her with either a random value or a value computed honestly for sk_I. Similarly, if no password is recovered, Alice is given a random key. Alice aborts in $(U5)$ if the signature of the server is invalid. A corruption of the server before $(U5)$ is dealt with as in $\boldsymbol{G}_4$. This is indistinguishable from the former game due to the smoothness of the hash function.
- From $\boldsymbol{G}_6$, $\mathcal{S}$ simulates the server without using his password anymore. It aborts if the signature received from the client is invalid. Otherwise, the server is given a random key

in ($S4$) in the case where `flow-one` was oracle-generated and the client was not corrupted. A corruption of the client before ($S4$) is dealt with as in $\boldsymbol{G}_4$: $\mathcal{S}$ asks a `GoodPwd` query for Bob and provides him with either a random value or a value computed honestly for sk_J (if no password is recovered, Bob is given a random key). This is indistinguishable from the former game due to the pseudo-randomness of the hash function.

- From $\boldsymbol{G}_7$, in the case where `flow-one` was not oracle-generated, $\mathcal{S}$ extracts pw_I from com_I and proceeds as described in $\boldsymbol{G}_4$: it asks a `GoodPwd` query for Bob and provides him with either a random value or a value computed honestly for sk_J (if no password is recovered, Bob is given a random key). This is indistinguishable from the former game due to the smoothness of the hash function.
- Finally, the hybrid queries are replaced by the real ones in $\boldsymbol{G}_8$, which is shown to be indistinguishable to the ideal-world experiment.

Description of the simulator The description of our simulator is based on that of [CHK+05]. When initialized with security parameter k, the simulator first runs the key-generation algorithm of the encryption scheme $\mathcal{E}$, thus obtaining a pair $(\mathsf{sk}, \mathsf{pk})$. It also chooses at random m elements $(y_1, \ldots, y_m)$ in G and a randomness extractor $\mathsf{Extract}$. It then initializes the real-world adversary $\mathcal{A}$, giving it $(\mathsf{pk}, (y_1, \ldots, y_m), \mathsf{Extract})$ as common reference string.

From this moment on, the simulator interacts with the environment $\mathcal{Z}$, the functionality $\mathcal{F}_{pwKE}$ and its subroutine $\mathcal{A}$. For the most part, this interaction is implemented by the simulator $\mathcal{S}$ just following the protocol on behalf of all the honest players. The main difference between the simulated players and the real honest players is that $\mathcal{S}$ does not engage on a particular password on their behalf. However, if $\mathcal{A}$ modifies a `flow-one` or a `flow-two` message that is delivered to player P in session ssid, then $\mathcal{S}$ decrypts that ciphertext (using sk) and uses the recovered message pw in a `TestPwd` query to the functionality. If this is a correct guess, then $\mathcal{S}$ uses this password on behalf of P, and proceeds with the simulation. More details follow.

Corruptions. Since we consider adaptive corruptions, that can occur at any moment during the execution of the protocol, our simulator, given the password of a player, needs to be able to provide the adversary with an internal state consistent with any data already sent (without the knowledge of the player's password at that time). To handle such corruptions, the key point relies in the equivocable property of our commitment. More precisely, instead of committing to a particular password, the simulator commits to *all* passwords, being able in the end to open to any of them. In a nutshell, committing to all passwords means to simulate the commitment in such a way that encryptions of all x_i, corresponding to all pw_i, are sent instead of encryptions of 0 (see the "equivocability" part of Section B.9 for details). Recall that the values hk and hp do not depend on the password, so that it does not engage the player on any of them.

Session Initialization. When receiving a message $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, \mathrm{I}, \mathrm{J}, \texttt{role})$ from $\mathcal{F}_{pwKE}$, $\mathcal{S}$ starts simulating a new session of the protocol for party P_I, peer P_J, session identifier ssid, and common reference string $(\mathsf{pk}, (y_1, \ldots, y_m), \mathsf{Extract})$. We denote this session by $(P_\mathrm{I}, \mathrm{ssid})$. If $\mathsf{role} = \texttt{client}$, then $\mathcal{S}$ generates a `flow-one` message by committing to all the passwords, choosing a key pair $(\mathrm{SK}_\mathrm{I}, \mathrm{VK}_\mathrm{I})$ for a one-time signature scheme. It gives this message to $\mathcal{A}$ on behalf of $(P_\mathrm{I}, \mathrm{ssid})$.

If $(P_\mathrm{I}, \mathrm{ssid})$ gets corrupted at this stage, then $\mathcal{S}$ recovers his password pw_I and is able to open his commitment in such a way that it is a commitment on pw_I. It can thus provide $\mathcal{A}$ with consistent data.

Protocol Steps. Assume that $\mathcal{A}$ sends a message m to an active session of some party. If this message is formatted differently from what is expected by the session, then $\mathcal{S}$ aborts that

session and notifies $\mathcal{A}$. Otherwise, we have the following cases (where we denote a party in the client role as P_{I} and a party in the server role as P_{J}):

1. Assume that the session $(P_{\mathrm{J}}, \mathrm{ssid})$ receives a message $m = (\texttt{flow-one}, \mathsf{com}_{\mathrm{I}}, \mathrm{VK}_{\mathrm{I}})$. Then, P_{J} is necessarily a server and m is the first message received by P_{J}. If $\mathsf{com}_{\mathrm{I}}$ is not equal to any commitment that was generated by $\mathcal{S}$ for a `flow-one` message, $\mathcal{S}$ uses its secret key sk to decrypt the ciphertext and obtain pw_{I} or nothing. Obtaining nothing is considered similar to an invalid password below due to the construction of the smooth hash function related to the commitment. When the extraction succeeds, because of the binding property, only one pw_{I} is possible (on-line dictionary attack), then $\mathcal{S}$ makes a call $(\texttt{TestPwd}, \mathrm{sid}, \mathrm{ssid}, \mathrm{J}, \mathsf{pw}_{\mathrm{I}})$ to the functionality. If this is a correct guess, then $\mathcal{S}$ sets the password of this server session to pw_{I}, otherwise, this is an invalid password. In both cases, $\mathcal{S}$ produces a commitment $\mathsf{com}_{\mathrm{J}}$ on all the passwords (it makes use of the equivocable property), chooses a key pair $(\mathrm{SK}_{\mathrm{J}}, \mathrm{VK}_{\mathrm{J}})$ for a one-time signature scheme, runs the key generation algorithms of the smooth hash function on $\mathsf{com}_{\mathrm{I}}$ to produce $(\mathsf{hk}_{\mathrm{J}}, \mathsf{hp}_{\mathrm{J}})$ and sends the `flow-two` message $(\mathsf{com}_{\mathrm{J}}, \mathsf{VK}_{\mathrm{J}}, \mathsf{hp}_{\mathrm{J}})$ to $\mathcal{A}$ on behalf of $(P_{\mathrm{J}}, \mathrm{ssid})$.

   If the sender $(P_{\mathrm{I}}, \mathrm{ssid})$ of this message, or if $(P_{\mathrm{J}}, \mathrm{ssid})$ gets corrupted at the end of this step, $\mathcal{S}$ handles this corruption just as when this player gets corrupted at the end of the initialization step. Note in addition that $\mathcal{S}$ is able to compute and give to $\mathcal{A}$ a correct value for $\mathsf{Hash}_{\mathrm{J}}$, the projected key being independent of the password (see discussion in Section B.9.3).

2. Assume that a session $(P_{\mathrm{I}}; \mathrm{ssid})$ receives a message $m = (\texttt{flow-two}, \mathsf{com}_{\mathrm{J}}, \mathsf{VK}_{\mathrm{J}}, \mathsf{hp}_{\mathrm{J}})$. Then, P_{I} must be a client who sent a `flow-one` message and is now waiting for the response. We say that $(P_{\mathrm{J}}, \mathrm{ssid})$ is a peer session to $(P_{\mathrm{I}}, \mathrm{ssid}')$ if $\mathrm{ssid} = \mathrm{ssid}'$, if session $(P_{\mathrm{I}}, \mathrm{ssid})$ has peer P_{J}, session $(P_{\mathrm{J}}, \mathrm{ssid})$ has peer P_{I}, and these two sessions have opposite roles (`client` /`server`). If the pair $(\mathsf{com}_{\mathrm{J}}, \mathsf{VK}_{\mathrm{J}})$ is not equal to the pair $(\mathsf{com}_{\mathrm{J}}, \mathsf{VK}_{\mathrm{J}})$ that was generated by $\mathcal{S}$ for a `flow-one` message from peer session $(P_{\mathrm{J}}, \mathrm{ssid})$ (or if no such ciphertext was generated yet, or no such peer session exists) then $\mathcal{S}$ uses its secret key sk to compute pw_{J}, or nothing which is considered similar to an invalid password below (as before). Also note that $\mathcal{S}$ can recover pw_{J} if $(P_{\mathrm{J}}, \mathrm{ssid})$ has been corrupted after having sent its commitment. In case of recovery of pw_{J}, $\mathcal{S}$ then makes a call $(\texttt{TestPwd}, \mathrm{sid}, \mathrm{ssid}, \mathrm{I}, \mathsf{pw}_{\mathrm{J}})$ to the functionality. If this is a correct guess, $\mathcal{S}$ sets the password of this client session to pw_{J}, otherwise, this is an invalid password.

   Then, it runs the key generation algorithms of the smooth hash function on $\mathsf{com}_{\mathrm{J}}$ to produce $(\mathsf{hk}_{\mathrm{I}}, \mathsf{hp}_{\mathrm{I}})$, as well as the signing algorithm with SK_{I} to compute σ_{I} and sends the `flow-three` message $(\sigma_{\mathrm{I}}, \mathsf{hp}_{\mathrm{I}})$ to $\mathcal{A}$ on behalf of $(P_{\mathrm{I}}, \mathrm{ssid})$.

   Note that in the former case (correct password guess), $\mathcal{S}$ computes honestly the session key using password pw_{J}, without issuing it yet. Otherwise, P_{I} is provided with a key chosen at random.

   If $(P_{\mathrm{J}}, \mathrm{ssid})$ gets corrupted at the end of this step, $\mathcal{S}$ handles this corruption just as when this player gets corrupted at the end of the previous step.

   If it is $(P_{\mathrm{I}}, \mathrm{ssid})$ that gets corrupted, we face two cases. If a correct password guess occurred in this step, then $\mathcal{S}$ has computed everything honestly and can provide every value to $\mathcal{A}$ (recall that the projection key does not depend on the password). Otherwise, if $\mathcal{S}$ has set the session key at random, recall that it has not sent anything yet, so that the adversary totally ignores the values computed. $\mathcal{S}$ then recovers the password of $(P_{\mathrm{I}}, \mathrm{ssid})$ and is able to compute the data and give them to the adversary.

3. Assume that a session $(P_{\mathrm{J}}, \mathrm{ssid})$ receives a message $m = (\texttt{flow-three}, \sigma_{\mathrm{I}}, \mathsf{hp}_{\mathrm{I}})$. Then, $(P_{\mathrm{J}}, \mathrm{ssid})$ must be a server who sent a `flow-two` message and is now waiting for the response. $\mathcal{S}$ aborts if the signature σ_{I} is not valid. If `flow-one` was not oracle-generated, then $\mathcal{S}$ has extracted the password pw_{I} from the commitment (or failed to extract it). Similarly, if the peer session P_{I} (exists and) was corrupted sooner in the protocol, then $\mathcal{S}$ knows its password pw_{I} (which was in particular used in the commitment). In both cases, the simulator makes a call $(\texttt{TestPwd}, \mathrm{sid}, \mathrm{ssid}, \mathrm{J}, \mathsf{pw}_{\mathrm{I}})$ to the functionality to check the compatibility of the two passwords. In case of a correct answer, $\mathcal{S}$ sets the password of this server session to pw_{I}, and computes honestly the session key using password pw_{I}. Otherwise, P_{J} is provided with a key chosen at random.

   Next, $\mathcal{S}$ runs signing algorithm with SK_{J} to compute σ_{J} and sends this `flow-four` message to $\mathcal{A}$ on behalf of $(P_{\mathrm{J}}, \mathrm{ssid})$.

   $\mathcal{S}$ handles a corruption of $(P_{\mathrm{I}}, \mathrm{ssid})$ just as it did at the end of the former step. And recall that no more corruption of $(P_{\mathrm{J}}, \mathrm{ssid})$ can occur since it claimed its session as "completed" and erased its data.

4. Assume that a session $(P_{\mathrm{I}}, \mathrm{ssid})$ receives a message $m = (\texttt{flow-four}, \sigma_{\mathrm{J}})$. $(P_{\mathrm{I}}, \mathrm{ssid})$ must then be a client who sent a `flow-three` message and is now waiting for the response. $\mathcal{S}$ aborts if the signature σ_{J} is not valid. If `flow-two` was not oracle-generated, then $\mathcal{S}$ has extracted the password pw_{J} from the commitment (or failed to). Similarly, if the peer session P_{J} (exists and) was corrupted sooner in the protocol, then $\mathcal{S}$ knows its password pw_{J} (which was in particular used in the commitment). In both cases, the simulator makes a call $(\texttt{TestPwd}, \mathrm{sid}, \mathrm{ssid}, \mathrm{J}, \mathsf{pw}_{\mathrm{J}})$ to the functionality to check the compatibility of the two passwords. In case of a correct answer, $\mathcal{S}$ sets the password of this client session to pw_{J} and computes honestly its session key. Otherwise, it sets its session key at random. Finally note that no corruption can occur at this stage.

If a session aborts or terminates, $\mathcal{S}$ reports it to $\mathcal{A}$. If the session terminates with a session key sk, then $\mathcal{S}$ makes a `NewKey` call to $\mathcal{F}_{pwKE}$, specifying the session key. But recall that unless the session is compromised or corrupted, $\mathcal{F}_{pwKE}$ will ignore the key specified by $\mathcal{S}$, and thus we do not have to bother with the key in these cases.

Description of the games We now provide the complete proof by a sequence of games. The detailed proof of some gaps are provided in Appendix B.11.

Game G_0: G_0 is the real game.

Game G_1: From this game on, we allow the simulator to program the common reference string, allowing it to know the trapdoors for extractability and equivocability.

Game G_2: This game is almost the same as the previous one. The only difference is that $\mathcal{S}$ always tries to extract the password committed to by the adversary (without taking advantage of the knowledge of this password for the moment) whenever the latter attempts to impersonate one of the parties. We allow the simulator to abort whenever this extraction fails because the adversary has generated a commitment which is valid for two or more passwords. Due to the binding property of the commitment (see Section B.9), the probability that the adversary achieves this is negligible. Note that the extraction can also fail if the values sent were not encryptions of 0 or 1 but we do not abort in this case; For the moment we still assume that the simulator knows the passwords of the players. In the following games, when it will not have this

knowledge anymore, we will show that the smooth hashes will be random so that this failure will have no bad consequences. This shows that $\boldsymbol{G}_2$ and $\boldsymbol{G}_1$ are indistinguishable.

Game $\boldsymbol{G}_3$: In this game, $\mathcal{S}$ still knows the passwords of both players, but it starts simulating the commitments by committing to all possible passwords (in order to be able to equivocate afterwards, see Section B.9 for details). Note that since the commitment is hiding, this does not change the view of an environment, and that the commitment remains binding (even under access to equivocable commitments —see Section B.9 and Appendix B.8). Also note that the generation of the projected keys for the smooth hash function (see Section B.9) is done without requiring the knowledge of the password. Hence, $\boldsymbol{G}_3$ and $\boldsymbol{G}_2$ are indistinguishable.

As a side note, we have just proven that the knowledge of the passwords is not necessary anymore for steps $(U1)$ and $(S2)$. If a player gets corrupted, the simulator recovers its password and is able to equivocate the commitment and thus provide the adversary with consistent data (since the projected key for the smooth hash function does not depend on the password committed).

Game $\boldsymbol{G}_4$: In this game, we suppose that `flow-one` was oracle-generated. We are now at beginning of round $(U3)$ and we want to simulate the (honest) client. We suppose that the simulator still knows the password of the server but not that of the client anymore. Let's first consider the case in which Alice received a `flow-two` which was oracle-generated. In such a case, the simulator chooses a random value $\mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{J})$. Then, if the server remains honest until $(S4)$, the simulator asks a `SamePwd` query to the functionality. If the answer is yes (that is $\mathsf{pw}_\mathrm{I} = \mathsf{pw}_\mathrm{J}$), it gives Bob the same random value for $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J}; \mathsf{w}_\mathrm{J})$ and computes honestly $\mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$, thus completely determining sk_J. Otherwise, it computes correctly the entire key. If both remain honest until $(U5)$, then, if they have the same password, $\mathcal{S}$ sets $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{I}; \mathsf{w}_\mathrm{I}) = \mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$. Otherwise, $\mathcal{S}$ sets sk_I at random.

The description of the corruptions and the proof of the indistinguishability between $\boldsymbol{G}_4$ and $\boldsymbol{G}_3$ can be found in Appendix B.11.1 (it follows from the pseudo-randomness of the smooth hash function).

Game $\boldsymbol{G}_5$: In this game, we still suppose that `flow-one` was oracle-generated, but we now consider the case in which `flow-two` was non-oracle-generated. We are now at beginning of round $(U3)$ and we want to simulate the (honest) client. The simulator still knows the password of the server.

com_J, VK_J or hk_J has been generated by the adversary. If com_J is a replay, then the adversary could not modify VK_J, because of the label used for com_J, and then the signature verification will fail. com_J is thus necessarily a new commitment. Recall that from $\boldsymbol{G}_2$, the simulator extracts the password from the commitment of the server, with the help of the secret key. The extraction can fail with negligible probability if $\mathcal{S}$ recovers two different passwords (thanks to the binding property, see $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$). In such a case, the simulator aborts the game. It can also happen that the extraction issues no password or that it fails if the values sent were not encryptions of 0 or 1.

Then, if it has recovered a password, the simulator asks a `GoodPwd` query to the functionality. If it is correct, then it computes honestly the session key of the client. Otherwise, or if it has not recovered any password in the extraction, it sets the value $\mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{J})$ at random (the entire key sk_I will be set in round $(U5)$). Note that the smooth hash value will necessarily be random by construction if the values sent in the commitment were not encryptions of 0 or 1.

The corruptions which can follow this step are dealt with as in the former game (recall that the projected keys sent do not depend on the password). Due to the smoothness of the hash

function, this game is indistinguishable from the previous one (see Appendix B.11.2).

Game $\boldsymbol{G}_6$: In this game, we deal with the case in which all flows received by the client up to $(S4)$ were oracle-generated. We are now at beginning of round $(S4)$ and we want to simulate the (honest) server. We suppose that the simulator doesn't know any passwords anymore. Let's first consider the case in which `flow-three` was oracle-generated. Note that `flow-one` must have been oracle-generated too, otherwise the signature σ_1 would have been rejected.

Then, the simulator asks a `SamePwd` query to the functionality. If it is correct, it sets the value for $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J}; \mathsf{w}_\mathrm{J})$ equal to $\mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J})$, and it also sets the value for $\mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$ at random. Otherwise, it sets these values at random.

In $(U5)$, if the passwords of Alice and Bob are the same and both remain honest, we set the value $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{I}; \mathsf{w}_\mathrm{I}) = \mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$: sk_I and sk_J are thus equal, as required. Note that since $\mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{J})$ is already set at random since $\boldsymbol{G}_4$ or $\boldsymbol{G}_5$, all the keys are already random.

If their passwords are not the same but both remain honest, Alice will be given in $(U5)$ a key chosen independently at random: Here, $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{I}; \mathsf{w}_\mathrm{I})$ doesn't have to be programmed, since the keys do not have any reason to be identical. In this case, as in $\boldsymbol{G}_4$, the pseudo-randomness of the hash functions ensures the indistinguishability (see Appendix B.11.3 for the treatment of corruptions).

Game $\boldsymbol{G}_7$: In this game, we still deal with the case in which all flows received by the client up to $(S4)$ were oracle-generated. We are now at beginning of $(S4)$ and we want to simulate the (honest) server, without knowing any password, but we now suppose that `flow-three` was not oracle-generated. Note that in this case `flow-one` cannot have been oracle-generated. Otherwise, the signature σ_1 would have been rejected.

Recall that from $\boldsymbol{G}_2$, the simulator extracts the password from the commitment of the client, with the help of the secret key. The extraction can fail with negligible probability if $\mathcal{S}$ recovers two different passwords (thanks to the binding property, see $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$). In such a case, the simulator aborts the game. It can also happen that the extraction issues no password (for example if the values sent in the commitment were not encryptions of 0 or 1). Then, if it has recovered a password, the simulator asks a `GoodPwd` query to the functionality. If it is correct, then it computes honestly the session key of the server. Otherwise, or if it has not recovered any password in the extraction, it sets the values $\mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$ and $\mathsf{ProjHash}(\mathsf{hp}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J}; \mathsf{w}_\mathrm{J})$ at random. Recall that if the values sent in the commitment were not encryptions of 0 or 1, the smooth hash value will be random.

The corruptions which can follow this step are dealt with as in the former game. Due to the smoothness of the hash function, this game is indistinguishable from the previous one: The proof is exactly the same as in $\boldsymbol{G}_5$, but it is made more easier, since the key of the client is already random.

Game $\boldsymbol{G}_8$: In this game, we replace `GoodPwd` queries by `TestPwd` ones, and `SamePwd` by `NewKey` ones. If a session aborts or terminates, $\mathcal{S}$ reports it to $\mathcal{A}$. If a session terminates with a session key sk, then $\mathcal{S}$ makes a `NewKey` call to the functionality, specifying the session key sk. But recall that unless the session is compromised, the functionality will ignore the key specified by $\mathcal{S}$.

We show in Appendix B.11.4 that this game is indistinguishable from the ideal-world experiment.

B.11 Appendix: Additional Proof Details

B.11.1 Indistinguishability of G_4 and G_3

We now describe what happens in case of corruptions. First, if Alice gets corrupted after the execution of $(U3)$, then the simulator will recover her password and thus be able to compute everything correctly. Recall that the projection key did not depend on the password.

Second, if Bob gets corrupted after $(U3)$ or after $(S4)$, then the simulator will be able to compute everything correctly. In particular in the second case, the adversary will get a coherent value of sk_J. Then, the simulator asks a `GoodPwd` query for Alice with Bob's password to the functionality. If they are the same, $\mathcal{S}$ recovers the password of Alice, and since it hasn't really erased her data, it will be able to compute sk_I exactly as the adversary should have done it for Bob (in particular because the projection key does not depend on the password). Otherwise, it gives Alice a random key: there is no need that the players get the same key if they don't share the same password – recall that all private data is erased so that the adversary cannot verify anything.

Finally, if Alice gets corrupted by the end of $(U5)$, the simulator will recover her password and ask a `SamePwd` query to the functionality. If σ_J is correct and they share the same password, then the simulator computes sk_I exactly as the adversary should have done it for Bob (one again because the projection key does not depend on the password). Otherwise, if the signature is correct but their passwords are different, then Alice is provided with a random value (recall that her data is supposed to be erased, so that the adversary is not supposed to be able to verify it). Otherwise, if the signature is incorrect, the simulator aborts the game. We can see here the necessity of step $(U5)$ in order to guarantee adaptive security.

We now show that an environment that distinguishes $\boldsymbol{G}_4$ from $\boldsymbol{G}_3$ can be used to construct a distinguisher between `Expt-Hash` and `Expt-Unif` as described in [GL03] (see Appendix B.6), violating their Corollary 3.3.

We first define hybrid games H_i as follows. First note that the sessions are ordered with respect to the rounds $(U1)$. In all sessions before the i^{th} one, the computation is random, and in all the sessions afterwards $(i, i+1, \ldots)$, the values are set as real. In all "random" cases, we set $\mathsf{ProjHash}(\mathsf{hp}_I; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J}; \mathsf{w}_J)$ to the same value during the simulation of the server if everything goes well, that is, if no corruption occurred and the passwords are the same. With these notations, the $i = 1$ case is exactly $\boldsymbol{G}_3$ and $i = q_s + 1$ is exactly $\boldsymbol{G}_4$. Pictorially,

$$\begin{cases} \text{Random} & 1, \ldots, i-1 \\ \text{Real} & i, \ldots, q_s \end{cases}$$

Our aim is now to prove the indistinguishability of H_i and H_{i+1}. We define the event E_i, stating that there exists a corruption between the committing $(U1)$ and hashing $(U3)$ steps in the i^{th} session. Notice that the probability of this event is the same in the two following games H_i and H_{i+1}. This is true despite the fact that we consider concurrent sessions. To see this, notice that even though E_i may depend on sessions with a larger index, the only difference between these two games concerns round $(U3)$ of the i^{th} session: everything is identical before this step. Since the corruption occurs before this round, the probability that the adversary corrupts a player in the i^{th} session, which only depends on what happened before, is the same in both cases.

We now denote by $\mathrm{OUT}_\mathcal{Z}$ the output of the environment at the end of the execution and compute the difference between $\Pr[\mathrm{OUT}_\mathcal{Z} = 1]$ in the two different games:

$$
\begin{aligned}
&\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1]\right| \\
=\ &\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1\wedge E_i]+\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1\wedge\neg E_i]\right. \\
&\qquad \left.-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1\wedge E_i]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1\wedge\neg E_i]\right| \\
\leq\ &\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1\wedge E_i]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1\wedge E_i]\right| \\
&\qquad +\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1\wedge\neg E_i]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1\wedge\neg E_i]\right| \\
\leq\ &\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1|E_i]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1|E_i]\right|\ \left|\Pr_{H_i}[E_i]\right| \\
&\qquad +\left|\Pr_{H_{i+1}}[\text{OUT}_{\mathcal{Z}}=1|\neg E_i]-\Pr_{H_i}[\text{OUT}_{\mathcal{Z}}=1|\neg E_i]\right|\ \left|\Pr_{H_i}[\neg E_i]\right|
\end{aligned}
$$

First consider the first term of this sum. If there is a corruption, we learn the password and also (in fact, we simulate in a coherent way) the randomness, enabling us to compute everything correctly. Thus, this term is equal to zero.

We now consider the second term, corresponding to the case where there is no corruption, and show that $|\Pr[\texttt{Expt-Hash}(D)=1]-\Pr[\texttt{Expt-Unif}(D)=1]|$ bounds (to within a negligible amount) the probability that the environment distinguishes H_{i+1} from H_i. More precisely, let's consider the following game H (as described below) in which the oracles `Commit` and `Hash` appear in the i^{th} session only under the assumption $\neg E_i$.

H is as follows: Let D be a machine that receives a randomly chosen public key pk_2 and emulates the game H_i with the following changes for the i^{th} session. On receiving a valid oracle-generated `flow-one`, D does not directly compute c_2 but it queries instead the oracle `Commit`() and sets c_2 to the value returned. If Alice receives the unmodified com_{J} in the `flow-two` message, then D queries $\texttt{Hash}(\mathsf{com}_{\text{J}})$ and receives a pair (s_I,η_I). Then it sets $\mathsf{hp}_I=s_I$ and $\mathsf{ProjHash}(\mathsf{hp}_I;\rho,(\ell_{\text{J}},\mathsf{pw}_{\text{I}}),\mathsf{com}_{\text{J}};\mathsf{w}_J)$. Note that w_J was here the randomness used by the oracle `Commit` in the query that generated com_{J}. Then, if Bob receives the unmodified projected key hp_I, D also uses η_I for the appropriate portion of the session key – in the case they have the same password. Finally, D outputs whatever the environment outputs. It is easy to see that $H=H_{i+1}$ in the case where the oracle `Hash` returns a random value, and it is equal to H_i otherwise. □

B.11.2 Indistinguishability of $\mathbf{G_5}$ and $\mathbf{G_4}$

First note that the corruptions which can follow this step are dealt with as in the former game, and that this simulation is compatible with the former game.

If the password recovered from the server is correct, the simulation is done honestly, so that this game is perfectly equivalent to the previous one. Otherwise, if the password is incorrect, or if no password was recovered, then com_{J} is invalid.

Given an invalid com_{J}, with $(P_{\text{I}},\text{ssid})$'s password pw_{I} and the label ℓ_{I}, the distribution $\{\mathsf{pk}_{\text{I}}, \mathsf{com}_{\text{I}},\ell_{\text{I}},\mathsf{pw}_{\text{I}},\mathsf{hp}_{\text{I}},\mathsf{Hash}(\mathsf{hk}_{\text{I}};\rho,(\ell_{\text{J}},\mathsf{pw}_{\text{I}}),\mathsf{com}_{\text{J}})\}$ is statistically close to the distribution $\{\mathsf{pk}_{\text{I}},\mathsf{com}_{\text{I}},\ell_{\text{I}},\mathsf{pw}_{\text{I}},\mathsf{hp}_{\text{I}},z\}$ where z is a random element of the group G (due to the smoothness of the hash function). Since $\mathsf{Hash}(\mathsf{hk}_{\text{I}};\rho,(\ell_{\text{J}},\mathsf{pw}_{\text{I}}),\mathsf{com}_{\text{J}})$ is a component of the session key sk_{I} for $(P_{\text{I}},\text{ssid})$, then the session key generated is statistically close to uniform.

Then, since com_{J} is invalid, $(P_{\text{J}},\text{ssid})$ will compute honestly the key, but he will not obtain the same session key since the passwords are different. This behavior is equivalent to what happens in the ideal functionality: the corresponding sessions of $(P_{\text{I}},\text{ssid})$ and $(P_{\text{J}},\text{ssid})$ either do not have a matching conversation, or were given different passwords by the environment, so that $(P_{\text{J}},\text{ssid})$ will not be given the same session key as $(P_{\text{I}},\text{ssid})$.

B.11.3 Indistinguishability of G_6 and G_5

The pseudo-randomness of the hash functions shows the indistinguishability between $\boldsymbol{G}_6$ and $\boldsymbol{G}_5$. Indeed, the environment cannot become aware that the keys were chosen at random. More precisely, we do the same manipulation as in the proof of $\boldsymbol{G}_4$, but this time considering the smooth projective hash function Hash with respect to $\mathsf{com}_{\mathrm{J}}$. Note that in the hybrid games, the sessions are ordered with respect to the rounds ($S2$).

We now consider the case where Bob gets corrupted between ($S4$) and ($U5$). Then, the simulator recovers his password and it is able to compute everything correctly (recall that the projection keys do not depend on the passwords). It then asks a `GoodPwd` query for the client. If their passwords are different, Alice is provided with a random key. Otherwise, $\mathcal{S}$ gives her the same key as Bob.

Finally note that for sake of simplicity, we only compute $\mathsf{Hash}_{\mathrm{J}}$ in ($S4$) in the simulation. But, if the server is corrupted after ($S2$), the simulator recovers his password and is able to provide the adversary with a correct $\mathsf{Hash}_{\mathrm{J}}$ (once more because the projection key does not depend on the password).

B.11.4 Indistinguishability between G_8 and the Ideal Game

The only difference between $\boldsymbol{G}_7$ and $\boldsymbol{G}_8$ is that the `GoodPwd` queries are replaced by `TestPwd` queries to the functionality and the `SamePwd` by a `NewKey` query. We say that the players have matching sessions if they share the same *ssid* and if they agree on the values of $\mathsf{com}_{\mathrm{I}}$, $\mathsf{com}_{\mathrm{J}}$, hp_I and hp_J (that is, all the values that determine the key). We now show that $\boldsymbol{G}_8$ and IWE are indistinguishable.

First, if both players share the same password and remain honest until the end of the game, and if there are no impersonations, they will obtain a random key, both in $\boldsymbol{G}_8$ (from $\boldsymbol{G}_6$) and IWE, as there are no `TestPwd` queries and the sessions remain fresh. Second, if they share the same password but there are impersonation attempts, then they receive independently-chosen random keys (from $\boldsymbol{G}_5$ or $\boldsymbol{G}_7$). Third, if they don't share the same password, then they get independently-chosen random keys. Now, we need to show that two players will receive the same key in $\boldsymbol{G}_8$ if and only if it happens in IWE.

First consider the case of players with matching session and the same password. If both players remain honest until the end of the game, they will receive the same key from $\boldsymbol{G}_6$. If not, it will still be the same from $\boldsymbol{G}_5$ or $\boldsymbol{G}_7$. Recall that if there are some impersonation attempts, the keys will be random and independent, both in $\boldsymbol{G}_8$ and IWE. In IWE, the functionality will receive two `NewSession` queries with the same password. If both are honest, it will not receive any `TestPwd` query, so that the key will be the same for the two players. And if one is corrupted and a `TestPwd` query is done (and correct, since they have the same password), then they will also have the same key, chosen by the adversary.

Then, consider the case of players with matching session but not the same password. If both players remain honest until the end of the game, they will receive independently-chosen random keys from $\boldsymbol{G}_6$. If not, it will still be the same from $\boldsymbol{G}_7$. In IWE, the functionality will receive two `NewSession` queries with different passwords. It will give them different keys by definition.

Finally, consider the case of players with no matching session. It is clear that in $\boldsymbol{G}_8$ the session keys of those players will be independent because they are not set in any of the games. In IWE, the only way that they receive matching keys is that the functionality receives two `NewSession` queries with the same password, and $\mathcal{S}$ sends a `NewKey` query for these sessions without having sent any `TestPwd` queries. But if the two sessions do not have a matching conversation, they must differ in either $\mathsf{com}_{\mathrm{I}}$, $\mathsf{com}_{\mathrm{J}}$, hp_I or hp_J. In this case, they will refuse the signature of the other player and abort the game.

If one of the players is corrupted by the end of the game, the simulator recovers its password and uses it in a `TestPwd` query for the other player to the functionality, as explained in $\boldsymbol{G}_4$. If the result is correct, then both players are give the same key. Otherwise, they are given independently-chosen random keys. This is exactly the behavior of the functionality in IWE.

Common reference string. A tuple $(G, \mathsf{pk}, (y_1, \ldots, y_m), \mathsf{Extract})$, where G is a cyclic group, $y_1, \ldots, y_m$ random elements from this group, pk a public key for the Cramer-Shoup scheme, and $\mathsf{Extract}$ a randomness extractor.

Protocol steps.

1. When P_I is activated with input $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, \mathrm{I}, \mathrm{J}, \mathsf{pw}_\mathrm{I}, \texttt{role})$, we face two cases: If $\texttt{role} = \texttt{server}$, it does nothing. If $\texttt{role} = \texttt{client}$, it uses SKG to generate a key pair $(\mathrm{VK}_\mathrm{I}, \mathrm{SK}_\mathrm{I})$ for a one-time signature scheme, sets the (public) label $\ell_\mathrm{I} = \mathrm{J} \circ \mathrm{I} \circ \mathrm{ssid} \circ \mathrm{VK}_\mathrm{I}$, computes $\mathsf{com}_\mathrm{I} = \mathsf{com}_\rho(\ell_\mathrm{I}, \mathsf{pw}_\mathrm{I}; R_\mathrm{I})$ and sends the message $(\texttt{flow-one}, \mathsf{com}_\mathrm{I}, \mathrm{VK}_\mathrm{I})$ to P_J. From this point on, assume that P_I is a party activated with input $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, \mathrm{I}, \mathrm{J}, \mathsf{pw}_\mathrm{I}, \texttt{client})$ and that P_J is a party activated with input $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, \mathrm{I}, \mathrm{J}, \mathsf{pw}_\mathrm{J}, \texttt{server})$. Recall that P_I erases nearly all the randoms (in R_I) used in the computation of com_I (see Section B.9). More precisely, it only keeps from R_I the values present in the witness w_I that will be used in the computation of the smooth hash function.

2. When P_J receives a message $(\texttt{flow-one}, \mathsf{com}_\mathrm{I}, \mathrm{VK}_\mathrm{I})$, it (publicly) checks that com_I is well constructed. Otherwise, it aborts. Then it uses SKG to generate a key pair $(\mathsf{VK}_\mathrm{J}, \mathsf{SK}_\mathrm{J})$ for a one-time signature scheme, and HashKG to generate a key hk_J for the smooth projective hash function family $\mathcal{H}$ with respect to ρ. It sets the (public) label $\ell_\mathrm{J} = \mathrm{J} \circ \mathrm{I} \circ \mathrm{ssid} \circ \mathsf{VK}_\mathrm{J}$ and computes the projection $\mathsf{hp}_\mathrm{J} = \mathsf{ProjKG}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$. Next it computes $\mathsf{com}_\mathrm{J} = \mathsf{com}_\rho(\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}; R_\mathrm{J})$ and sends the message $(\texttt{flow-two}, \mathsf{com}_\mathrm{J}, \mathrm{VK}_\mathrm{J}, \mathsf{hp}_\mathrm{J})$ to P_I. It finally computes $\mathsf{Hash}_J = \mathsf{Hash}(\mathsf{hk}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{I})$ and erases hk_J and the values of R_J that are not present in the witness w_J.

3. When P_I receives a message $(\texttt{flow-two}, \mathsf{com}_\mathrm{J}, \mathrm{VK}_\mathrm{J}, \mathsf{hp}_\mathrm{J})$, it (publicly) checks that com_J is well constructed. Otherwise, it aborts. Then it uses HashKG to generate a key hk_I for the smooth projective hash function family $\mathcal{H}$ with respect to pk and computes the projection $\mathsf{hp}_\mathrm{I} = \mathsf{ProjKG}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{J})$. Next it computes $\sigma_\mathrm{I} = \mathsf{Sign}(\mathrm{SK}_\mathrm{I}, (\mathsf{com}_\mathrm{I}, \mathsf{com}_\mathrm{J}, \mathsf{hp}_\mathrm{I}, \mathsf{hp}_\mathrm{J}))$ and sends the message $(\texttt{flow-three}, \sigma_\mathrm{I}, \mathsf{hp}_\mathrm{I})$ to P_J. It computes the session key $sk_\mathrm{I} = \mathsf{ProjHash}(\mathsf{hp}_\mathrm{J}; \rho, (\ell_\mathrm{I}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{I}; \mathsf{w}_\mathrm{I}) + \mathsf{Hash}(\mathsf{hk}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{I}), \mathsf{com}_\mathrm{J})$, and erases all private data except pw_I and sk_I, keeping all public data in memory.

4. When P_J receives a message $(\texttt{flow-three}, \sigma_\mathrm{I}, \mathsf{hp}_\mathrm{I})$, it checks that $\mathsf{Ver}(\mathrm{VK}_\mathrm{I}, (\mathsf{com}_\mathrm{I}, \mathsf{com}_\mathrm{J}, \mathsf{hp}_\mathrm{J}), \sigma_\mathrm{I}) = 1$. If not, it aborts the session outputting nothing. Otherwise, it computes $\sigma_\mathrm{J} = \mathsf{Sign}(\mathrm{SK}_\mathrm{J}, (\mathsf{com}_\mathrm{I}, \mathsf{com}_\mathrm{J}, \mathsf{hp}_\mathrm{I}, \mathsf{hp}_\mathrm{J}))$ and sends the message $(\texttt{flow-four}, \sigma_\mathrm{J})$ to P_I. Then, it computes the session key $\mathsf{sk}_\mathrm{J} = \mathsf{Hash}_\mathrm{J} + \mathsf{ProjHash}_\mathrm{I}(\mathsf{hp}_\mathrm{I}; \rho, (\ell_\mathrm{J}, \mathsf{pw}_\mathrm{J}), \mathsf{com}_\mathrm{J}; \mathsf{w}_\mathrm{J})$, outputs $(\mathrm{sid}, \mathrm{ssid}, \mathsf{sk}_\mathrm{J})$, sets the session as $\texttt{accepted}$, which means in particular that it erases everything except pw_J and sk_J and then terminates (i.e., publishes the session key).

5. When P_I receives a message $(\texttt{flow-four}, \sigma_\mathrm{J})$, it checks that $\mathsf{Ver}(\mathsf{VK}_\mathrm{J}, (\mathsf{com}_\mathrm{I}, \mathsf{com}_\mathrm{J}, \mathsf{hp}_\mathrm{I}, \mathsf{hp}_\mathrm{J}), \sigma_\mathrm{J}) = 1$. If not, it aborts the session outputting nothing. Otherwise, it terminates the session (i.e., publishes the session key).

Figure B.3: Description of the protocol for two players Alice and Bob. Alice is the client P_I, with index I and password pw_I, and Bob is the server P_J, with index J and password pw_J.

Appendix C

Password-based Group Key Exchange in a Constant Number of Rounds

PKC 2006

[ABCP06] with E. Bresson, O. Chevassut, and D. Pointcheval

Abstract : *With the development of grids, distributed applications are spread across multiple computing resources and require efficient security mechanisms among the processes. Although protocols for authenticated group Diffie-Hellman key exchange protocols seem to be the natural mechanisms for supporting these applications, current solutions are either limited by the use of public key infrastructures or by their scalability, requiring a number of rounds linear in the number of group members. To overcome these shortcomings, we propose in this paper the first provably-secure password-based constant-round group key exchange protocol. It is based on the protocol of Burmester and Desmedt and is* provably-secure *in the random-oracle and ideal-cipher models, under the Decisional Diffie-Hellman assumption. The new protocol is very efficient and fully scalable since it only requires four rounds of communication and four multi-exponentiations per user. Moreover, the new protocol avoids intricate authentication infrastructures by relying on passwords for authentication.*

C.1 Introduction

Motivation. Modern distributed applications often need to maintain consistency of replicated information and coordinate the activities of many processes. Collaborative applications and distributed computations are both examples of these types of applications. With the development of grids [FK04], distributed computations are spread across multiple computing resources requiring efficient security mechanisms between the processes. Although protocols for group Diffie-Hellman key exchange [BCP01, BCP02b, BCP02a, BCPQ01] provide a natural mechanism for supporting these applications, these protocols are limited in their scalability due to a number of rounds linear in the number of group members. An alternative is to use a protocol for group key exchange that runs in a constant number or rounds [DB06, KY03, KLL04]. The two measures of a protocol's efficiency are the computational cost per member and the communication complexity (number of protocol rounds) of the given protocol. Since the Moore's

laws has told us that computing power grows faster than communication power, it is therefore natural to trade communication power for computing power in a group key exchange protocol.

A password is the ideal authentication means to exchange a session key in the absence of public-key infrastructures or pre-distributed symmetric keys. In a group, the sharing of a password among the members greatly simplifies the setup of distributed applications [BCP02b, DB06]. An example of distributed applications could simply be the networking of all the devices attached to a human. Low-entropy passwords are easy for humans to remember, but cannot of course guarantee the same level of security as high-entropy secrets such as symmetric or asymmetric keys. The most serious attack against a password-based protocol is the so-called *dictionary attack*: the attacker recovers the password and uses it to impersonate the legitimate user. The low-entropy feature makes the job of the attacker easier since the attacker (off-line) runs through all the possible passwords in order to obtain partial information and to maximize his success probability. The minimum required from a protocol is security against this attack.

Contributions. In the present paper, we study the problem of scalable protocols for authenticated group Diffie-Hellman key exchange. Many researchers have studied and found solutions to this problem in the context of a Public-Key Infrastructure (PKI), yet a (secure) solution had to be found in the context of a (short) password shared among the members of the group. Two attempts in this direction are due to Dutta and Barua [DB06] and to Lee, Hwang, and Lee [LHL04]. Unfortunately, adding authentication services to a group key exchange protocol is a not trivial since redundancy in the flows of the protocol can open the door to different forms of attacks. In fact, in Section C.3, we briefly describe attacks against the schemes of Dutta and Barua [DB06] and of Lee, Hwang, and Lee [LHL04]. Then, in Section C.4, we show how to add password-authentication services to the Burmester and Desmedt scheme [BD94, BD05]. Our protocol is *provably secure* in the random-oracle [BR94b] and ideal-cipher models [BPR00] under the Decisional Diffie-Hellman assumption.

Related Work. Following the work of Bresson et al. on the group Diffie-Hellman key exchange problem [BCP01, BCP02b, BCP02a, BCPQ01], several researchers have developed similar protocols but that run in a constant number of rounds. Katz and Yung [KY03] added authentication services to the original Burmester and Desmedt's protocol [BD94, BD05]. Later, Kim, Lee and Lee extended the work of Katz and Yung to take into account the notion of dynamicity in the membership [KLL04]. The problem of adding password-authentication services followed shortly after. In [BCP02b], Bresson et al. proposed the first solution to the group Diffie-Hellman key exchange problem in the password-based scenario. Their protocol, however, has a total number of rounds which is linear in the total number of players in the group. In [DB06, LHL04], two different password-based versions of Burmester-Desmedt protocol were proposed along with proofs in the random-oracle and ideal-cipher models. Unfortunately, the latter two schemes are not secure.

Outline of the paper. The paper is organized as follows. In Section C.2, we recall the security model usually used for password-based group Diffie-Hellman key exchange. This model was previously defined in [BCP02b], but also takes advantage of [AFP05]. In Section C.3 we recall Burmester-Desmedt scheme and describe attacks against the schemes of Dutta and Barua [DB06] and of Lee, Hwang, and Lee [LHL04]. In Section C.4, we describe the mechanics behind our protocol. In Section C.5, we show that our protocol is *provably-secure* in the random-oracle and ideal-cipher models under the Decisional Diffie-Hellman assumption.

C.2 Security Model

C.2.1 Password-Based Authentication

In the password-based authentication setting, we assume each player holds a password pw drawn uniformly at random from the dictionary $\mathsf{Password}$ of size N. This secret of low-entropy (N is often assumed to be small, *i.e.* typically less than a million) will be used to authenticate the parties to each other

Unfortunately, one cannot prevent an adversary to choose randomly a password in the dictionary and to try to impersonate a player. However such on-line exhaustive search (even if N is not so large) can easily be *limited* by requiring a minimal time interval between successive failed attempts or locking an account after a threshold of failures. Security against such active attacks is measured in the number of passwords the adversary can "erase" from the candidate list after a failure.

On the other hand, off-line exhaustive search cannot be limited by such practical behaviors or computational resources considerations. Hopefully, they can be *prevented* if the protocol is carefully designed and ensures that no information about the password can leak from passively eavesdropped transcripts, but also from active attacks.

C.2.2 Formal Definitions

We denote by $U_1, \ldots, U_n$ the parties that can participate in the key exchange protocol P. Each of them may have several *instances* called oracles involved in distinct, possibly concurrent, executions of P. We denote U_i instances by U_i^j. The parties share a low-entropy secret pw which is uniformly drawn from a small dictionary $\mathsf{Password}$ of size N.

The key exchange algorithm P is an interactive protocol between the U_i's that provides the instances with a session key sk. During the execution of this protocol, the adversary has the entire control of the network, and tries to break the privacy of the key.

Remark C.2.1 In the "constant-round" protocols that we will study, simultaneous broadcasts are intensively used. However we do not make any assumption about the correctness of the latter primitive: it is actually a multi-cast, in which the adversary may delay, modify, or cancel the message sent to each recipient independently.

In the usual security model [BCP02b], several queries are available to the adversary to model his capability. We however enhance it with the Real-or-Random notion for the semantic security [AFP05] instead of the Find-then-Guess. This notion is strictly stronger in the password-based setting. And actually, since we focus on the semantic security only, we can assume that each time a player accepts a key, the latter is revealed to the adversary, either in a real way, or in a random one (according to a bit b). Let us briefly review each query:

- $\mathsf{Send}(U_i^j, m)$: This query enables to consider active attacks by having $\mathcal{A}$ sending a message to any instance U_i^j. The adversary $\mathcal{A}$ gets back the response U_i^j generates in processing the message m according to the protocol P. A query $\mathsf{Send}(\texttt{Start})$ initializes the key exchange algorithm, and thus the adversary receives the initial flows sent out by the instance.

- $\mathsf{Test}^b(U_i^j)$: This query models the misuse of the session key by instance U_i (*known-key attacks*). The query is only available to $\mathcal{A}$ if the attacked instance actually "holds" a session key. It either releases the actual key to $\mathcal{A}$, if $b = 1$ or a random one, if $b = 0$. The random keys must however be consistant between users in the same session. Therefore, a random key is simulated by the evaluation of a random function on the view a user has

of the session: all the partners have the same view, they thus have the same random key (but independent of the actual view.)

Remark C.2.2 Note that it has been shown [AFP05] that this query is indeed enough to model *known-key attacks* —where Reveal queries, which always answer with the real keys, are available—, and makes the model even stronger. Even though their result has only been proven in the two-party and three-party scenarios, one should note that their proof can be easily extended to the group scenario.

As already noticed, the aim of the adversary is to break the privacy of the session key (a.k.a., semantic security). This security notion takes place in the context of executing P in the presence of the adversary $\mathcal{A}$. One first draws a password pw from Password, flips a coin b, provides coin tosses to $\mathcal{A}$, as well as access to the Test^b and Send oracles.

The goal of the adversary is to guess the bit b involved in the Test queries, by outputting this guess b'. We denote the **AKE advantage** as the probability that $\mathcal{A}$ correctly guesses the value of b. More precisely we define $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$. The protocol P is said to be (t, ϵ)-**AKE-secure** if $\mathcal{A}$'s advantage is smaller than ϵ for any adversary $\mathcal{A}$ running with time t.

C.2.3 On the Simplification of the Model

In previous models, Execute queries were introduced to model passive eavesdropping. However, they can easily be simulated using the Send queries. In our analysis, we refine the way to deal with the adversary possible behaviors. We will denote by q_{active} the number of messages the adversary produced by himself (thus without including those he has just forwarded). This number upper-bounds the number of on-line "tests" the adversary performs to guess the password. And we denote by q_{session} the total number of sessions the adversary has initiated: nq_{session}, where n is the size of the group, upper-bounds the total number of messages the adversary has sent in the protocol (including those he has built and those he has just forwarded). We emphasize that this is stronger than considering only Execute and Send queries: while being polynomially equivalent, the two models are not tightly equivalent, since the adversary does not need to know in advance if he will forward all the flows, or be active when a new session starts. Moreover, suppressing the Execute queries makes the model even simpler.

The best we can expect with such a scheme is that the adversary erases no more than 1 password for each *session* in which he plays actively (since there exists attacks which achieve that in any password-based scheme.) However, in our quite efficient scheme, we can just prevent the adversary from erasing more than 1 password for each *player* he tries to impersonate (we will even show our proof is almost optimal.)

C.3 Preliminaries

The best starting point for an efficient password-based group key exchange, and namely if one wants a constant-round protocol, is the scheme proposed by Burmester and Desmedt [BD94, BD05] at Eurocrypt 94 and later formally analyzed by Katz and Yung in 2003 [KY03].

C.3.1 The Burmester and Desmedt Protocol

In the Burmester-Desmedt scheme, one considers a cyclic group $\mathbb{G}$ generated by g, in which the Decisional Diffie-Hellman (DDH) assumption holds. The protocol works as follows, where all the indices are taken modulo n (between 1 and n), and n is the size of the group:

- Each player U_i chooses a random exponent x_i and broadcasts $z_i = g^{x_i}$;
- Each player computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$, and broadcasts $X_i = Z_{i+1}/Z_i$;
- Each player computes his session key as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$.

It is easy to see that for any i, we have $K_i = \prod_{j=1}^{j=n} Z_j = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$.

C.3.2 A Naive Password-Based Approach

We immediately note that encrypting values in the second round would lead to a trivial dictionary attack, since the product of all values is equal to 1. One may want to enhance the Burmester and Desmedt's protocol by using a password pw to "mask" the first round only. One then comes up to the simple protocol, using a mask of the form h^{pw}, where h is another generator of the group $\mathbb{G}$, whose discrete logarithm in the base g is unknown [AP05]:

- Each player U_i chooses a random exponent x_i, computes $z_i = g^{x_i}$ and broadcasts $z_i^\star = z_i h^{pw}$;
- Each player extracts z_{i-1} and z_{i+1}, and computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$. He then broadcasts $X_i = Z_{i+1}/Z_i$;
- Each player computes his secret as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$

Thereafter, one can add any key confirmation and/or any intricate key extraction (even in the random oracle model, such as $sk_i = \mathcal{H}(\mathsf{View}, K_i)$), but it does not help. Indeed, the homomorphic property of this "masking" technique allows active attacks from the adversary: Assume that the adversary impersonates players U_1 and U_3 and sends for the first round $z_1^\star = g^{u_1}$ and $z_3^\star = g^{u_3}$, for known values u_1 and u_3. On the second round, the adversary waits for receiving X_2 from player U_2:

$$X_2 = \left(\frac{z_3}{z_1}\right)^{x_2} = g^{x_2(u_3-u_1)} = \left(\frac{z_2}{h^{pw}}\right)^{u_3-u_1}.$$

Then one knows that $h^{pw} = z_2/X_2^{(u_1-u_3)^{-1}}$, which can be easily checked off-line: a dictionary attack.

Furthermore, one can be easily convinced that any mechanism such as proof of knowledge, commitments, etc. to "enforce" the adversary to properly construct his values are useless against this attack, since in the above attack, the adversary plays "honestly".

C.3.3 The Dutta and Barua Protocol

Dutta and Barua [DB06] proposed a variant of the Kim-Lee-Lee protocol [KLL04] presented at Asiacrypt '04. It makes use of the ideal-cipher model, instead of a simple mask as above, and is claimed to be secure against dictionary attacks:

- Each player U_i chooses a random exponent x_i, as well as a random key k_i, computes $z_i = g^{x_i}$, and broadcasts $z_i^\star = \mathcal{E}_{pw}(z_i)$;
- Each player extracts z_{i-1} and z_{i+1}, and computes the $K_i^L = \mathcal{H}(z_{i-1}^{x_i}) = \mathcal{H}(g^{x_{i-1}x_i})$ and $K_i^R = \mathcal{H}(z_i^{x_{i+1}}) = \mathcal{H}(z_{i+1}^{x_i}) = \mathcal{H}(g^{x_ix_{i+1}})$. For $i = 1, \ldots, n-1$, U_i computes $X_i = K_i^L \oplus K_i^R$, while U_n computes $X_n = k_n \oplus K_n^R$; For $i = 1, \ldots, n-1$, U_i broadcasts $\mathcal{E}'_{pw}(k_i \| X_i)$, while U_n broadcasts $\mathcal{E}''(X_n)$;

- After decryption, they can all recover all the k_i, and then the common session key is set as $sk = \mathcal{H}(k_1 \| \ldots \| k_n)$.

Unfortunately, their protocol contains another source of redundancy that can be exploited by an attacker: the encryption algorithm of all users use the password as their encryption key. Therefore, a simple attack against their scheme runs as follows: the adversary plays the role of user U_3, with honest users U_1 and U_2. When the adversary receives $z_1^\star = \mathcal{E}_{pw}(z_1)$ and $z_2^\star = \mathcal{E}_{pw}(z_2)$, he sets $z_3^\star = Z_1^\star$, sends it to users U_1 and U_2, and waits for their responses. Note that setting $z_3^\star = Z_1^\star$ implicitly sets $x_3 = x_1$. At this point, the adversary knows that $K_2^L = \mathcal{H}(g^{x_1 x_2})$ and $K_2^R = \mathcal{H}(g^{x_2 x_3}) = \mathcal{H}(g^{x_1 x_2})$, and thus $X_2 = 0^k$ (where k is the output length of the function $\mathcal{H}$). Upon receiving $\mathcal{E}'_{pw}(k_2 \| X_2)$ from U_2, he can perform an off-line dictionary attack that immediately leads to the correct password, since this will be the only one decrypting this value to $k_2 \| 0^k$.

This confirms the fact that converting a provably-secure scheme into a password-based protocol is not a simple task. The main problem we observe with the above scheme is the unique way in which the initial messages of all users are encrypted, allowing attacks where one player can easily replay messages from another player. Thus, to avoid problems such as these, one should at least make sure that the encryption key used by each user is unique to that user. In fact, this is one of the features of the protocol that we present in the next section.

C.3.4 The Lee-Hwang-Lee Protocol

In [LHL04], Lee, Hwang, and Lee proposed another password-based version of the Burmester-Desmedt protocol, which makes use of the random-oracle and ideal-cipher models. Let $\mathcal{E}$ be an ideal cipher and let $\mathcal{H}$ and $\mathcal{H}'$ be random oracles. Their protocol works as follows:

- Each player U_i chooses a random exponent x_i, computes $z_i = g^{x_i}$, and broadcasts $(U_i, z_i^\star = \mathcal{E}_{pw}(z_i))$;

- Each player U_i extracts z_{i-1} and z_{i+1}, computes $K_i = \mathcal{H}(z_{i+1}^{x_i}) = \mathcal{H}(g^{x_i x_{i+1}})$, $K_{i-1} = \mathcal{H}(z_{i-1}^{x_i}) = \mathcal{H}(g^{x_{i-1} x_i})$, $w_i = K_{i-1} \oplus K_i$, and broadcasts (U_i, w_i).

- Each player U_i first computes the values $K_j = \mathcal{H}(g^{x_{j-1} x_j})$ for $j = 1, \ldots, n$, using the values w_j that were broadcasted in the second round. Next, each player U_i sets $sk = \mathcal{H}'(\mathcal{H}(g^{x_1 x_2}) \| \ldots \| \mathcal{H}(g^{x_{n-1} x_n}) \| \mathcal{H}(g^{x_n x_1}))$ as the common session key.

To show that the protocol above is not secure, we present the following simple attack against the semantic security of the session key. First, we start two sessions with player U_1 using $\{U_1, \ldots, U_4\}$ as the group. Let x_1 and x_1' be the corresponding values chosen by the two instances of player U_1 in each of these sessions and let $(U_1, z_1^\star = \mathcal{E}_{pw}(g^{x_1}))$ and $(U_1, z'^\star_1 = \mathcal{E}_{pw}(g^{x'_1}))$ be the corresponding values outputted by these instances. For the instance that outputted $(U_1, z_1^\star)$, we provide to it the values $(U_2, z'^\star_1)$, $(U_3, z_1^\star)$, and $(U_4, z'^\star_1)$, as the first-round messages of players U_2, U_3, and U_4. This implicitly makes $K_1 = K_2 = K_3 = K_4 = \mathcal{H}(g^{x'_1 x_1})$. Likewise, for the instance that outputted $(U_1, z'^\star_1)$, we provide to it the values $(U_2, z_1^\star)$, $(U_3, z'^\star_1)$, and $(U_4, z_1^\star)$, as the first-round messages of players U_2, U_3, and U_4. This implicitly makes $K_1' = K_2' = K_3' = K_4' = \mathcal{H}(g^{x'_1 x_1})$. As a result, $w_1 = w_2 = w_3 = w_4 = 0$ and $w_1' = w_2' = w_3' = w_4' = 0$ and, thus, we can easily compute the appropriate second-round messages for players U_2, U_3, and U_4 in both sessions. Moreover, the session keys of these two sessions are the same. Thus, we can ask test queries to both instances of player U_1 and check whether we get back the same value. This should be the case whenever the output of test oracle is the actual session key.

C.4 Our protocol

As above, we use the ideal-cipher model. The latter considers a family of random permutations $\mathcal{E}_k : \mathbb{G} \to \mathbb{G}$ indexed by a $\ell_{\mathcal{H}}$-bit key k which are accessible (as well as their inverses) through oracle queries ($\mathcal{E}$ and $\mathcal{D}$). Here we use the password, together with nonces, and the index of the user, to encrypt the values in the first round. Other values are sent in the clear. Also a preliminary round is used during which each player chooses random nonces to be used. This will be crucial to define sessions, and then link the encrypted values altogether.

Key generations (for the symmetric encryption $\mathcal{E}$, and for the session key) will make use of hash functions $\mathcal{H} : \{0,1\}^\star \to \{0,1\}^{\ell_{\mathcal{H}}}$ and $\mathcal{G} : \{0,1\}^\star \to \{0,1\}^{\ell_{\mathcal{G}}}$. Key confirmations will apply the function $\mathcal{A}uth : \{0,1\}^\star \to \{0,1\}^{\ell_{\mathcal{A}uth}}$.

C.4.1 Description

The protocol runs as follows:

1. Each player U_i chooses a random nonce N_i and broadcasts (U_i, N_i);
2. The session $S = U_1 \| N_1 \| \ldots \| U_i \| N_i \ldots \| U_n \| N_n$ is then defined, in which each player has a specific index i, and a specific symmetric key $k_i = \mathcal{H}(S, i, pw)$. Each player U_i chooses a random exponent x_i and broadcasts $z_i^\star = \mathcal{E}_{k_i}(z_i)$, where $z_i = g^{x_i}$;
3. Each player extracts $z_{i-1} = \mathcal{D}_{k_{i-1}}(z_{i-1}^\star)$ and $z_{i+1} = \mathcal{D}_{k_{i+1}}(z_{i+1}^\star)$, and computes the $Z_i = z_{i-1}^{x_i}$ and $Z_{i+1} = z_i^{x_{i+1}} = z_{i+1}^{x_i}$. He then broadcasts $X_i = Z_{i+1}/Z_i$;
4. Each player computes his secret as $K_i = Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}$, and broadcasts his key confirmation $\mathcal{A}uth_i = \mathcal{A}uth(S, \{z_j^\star, X_j\}_j, K_i, i)$.
5. After having received and checked all the key confirmations, each player defined is session key as $sk_i = \mathcal{G}(S, \{z_j^\star, X_j, \mathcal{A}uth_j\}_j, K_i)$.

C.4.2 Security Theorem

Here we present the main security result of this paper, whose proof appears in Section C.5.

Theorem C.4.1 Let P the above protocol in which the password is chosen in a dictionary of size N. Then for any adversary $\mathcal{A}$ running in time t, that makes at most q_{active} attempts within at most $q\mathit{Session}$ sessions, his advantage in breaking the semantic security of the session key, in the ideal-cipher model, is upper-bounded by:

$$\begin{aligned}\mathsf{Adv}_P^{\mathsf{ake}}(t) \;\leq\; & \frac{2q_{\mathsf{active}}}{N} + 4q_{\mathsf{session}} n \mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t) + \frac{2q_{\mathcal{G}}^2}{2^{\ell_{\mathcal{G}}}} + \frac{2q_{\mathcal{A}uth}^2}{2^{\ell_{\mathcal{A}uth}}} \\ & + \frac{8q_{\mathcal{G}} + 2q_{\mathcal{A}uth} + 2q_{\mathcal{D}} + 2nq_{\mathcal{E}}q_{\mathsf{session}} + (q_{\mathcal{E}} + q_{\mathcal{D}})^2}{|\mathbb{G}|} + \frac{2q_{\mathcal{H}}(q_{\mathcal{H}} + q_{\mathcal{D}})}{2^{\ell_{\mathcal{H}}}}\end{aligned}$$

where $q_{\mathcal{G}}, q_{\mathcal{H}}, q_{\mathcal{A}uth}, q_{\mathcal{E}}, q_{\mathcal{D}}$ denote the number of oracle queries the adversary is allowed to make to the random oracles $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{A}uth$, and to the ideal-cipher oracles $\mathcal{E}$ and $\mathcal{D}$, respectively.

This theorem states that the security of the session key is protected against dictionary attacks: the advantage of the adversary essentially grows linearly with the number of *active attempts* that the adversary makes (i.e., the number of messages that the adversary builds by himself). While the number of *sessions* includes both active attacks and passive ones (i.e., the session transcripts $\mathcal{A}$ passively eavesdropped), the theorem shows that these passive attacks are essentially negligible: a honest transcript does not help a computationally bounded adversary in guessing the password.

C.4.3 On the tightness of Theorem C.4.1

Clearly, Theorem C.4.1 ensures that when building a message by himself, the adversary cannot "test" more than one password per message. Actually, in the proof, we use q_{active} to upper-bound the number of players the adversary tries to impersonate and thus the number of different passwords he can inject. Hence, we achieve a stronger security result than the one claimed in Theorem C.4.1. However, it leaves open the possibility of whether an adversary can test several passwords in the same session. Since one may wonder whether a security proof with a tighter reduction could be found, here we present an online dictionary attack against our scheme that shows that this is not the case. More precisely, we exhibit an online dictionary attack in which the adversary can test several passwords in the same session (but still no more than one password for each message!). The idea behind the attack is to create a session in which the number of dishonest players (whose roles are played by the adversary) is twice the number of honest players and to surround each of the honest players with two dishonest players.

Let k be the number of honest players. The attack works as follows. First, the adversary starts a session in which all the honest players have indices of the form $3(i-1)+2$ for $i = 1, \ldots, k$. Then, let $\{pw_1, \ldots, pw_k\}$ be a list of candidate passwords that an adversary wants to try and let $i' = 3(i-1)$. To test whether pw_i for $i = 1, \ldots, k$ is the correct password, the adversary plays the role of players $U_{i'+1}$ and $U_{i'+3}$ and follows the protocol using pw_i as the password. That is, he chooses random exponents $x_{i'+1}$ and $x_{i'+3}$, computes the values $z_{i'+1} = g^{x_{i'+1}}$ and $z_{i'+3} = g^{x_{i'+3}}$, and then computes $z^{\star}_{i'+1}$ and $z^{\star}_{i'+3}$ from $z_{i'+1}$ and $z_{i'+3}$ using pw_i as the password. Let $X_{i'+2}$ be the value that the honest user $U_{i'+2}$ outputs in the third round of our protocol. To verify if his guess pw_i for the password is the correct one, the adversary computes $z_{i'+2}$ from $z^{\star}_{i'+2}$ using pw_i as the password and checks whether $z_{i'+2}^{x_{i'+3}-x_{i'+1}} = X_{i'+2}$. This should be the case whenever pw_i is equal to the actual password.

C.4.4 Computational Assumptions

Decisional Diffie-Hellman assumption: DDH. The DDH assumption states (roughly) that the distributions (g^u, g^v, g^{uv}) and (g^u, g^v, g^w) are computationally indistinguishable when u, v, w are indices chosen uniformly at random. This can be made more precise by defining two experiments, $\mathsf{DDH}^{\star}$ and $\mathsf{DDH}^{\$}$. In experiment $\mathsf{DDH}^{\star}$, the inputs given to the adversary are $U = g^u$, $V = g^v$, and $W = g^{uv}$, where u and v are two random indices. In experiment $\mathsf{DDH}^{\$}$, the inputs given to the adversary are $U = g^u$, $V = g^v$, and $W = g^w$, where u, v, and w are random indices. The goal of the adversary is to guess a bit indicating the experiment he thinks he is in. A (t, ϵ)-distinguisher against DDH for $\mathbb{G}$ is a probabilistic Turing machine Δ with time-complexity t, which is able to distinguish these two distributions with an advantage $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\Delta)$ greater than ϵ. The **advantage function** $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$ for the group $\mathbb{G}$ is then defined as the maximum value of $\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(\Delta)$ over all Δ with time-complexity at most t.

Parallel Decisional Diffie-Hellman assumption: PDDH. We define a variant of the DDH problem, we name it the *Parallel Decisional Diffie-Hellman* problem, which is equivalent to the usual DDH problem. To this aim, we define the two following distributions:

$$\mathsf{PDH}^{\star}_n = \{g^{x_1}, \ldots, g^{x_n}, g^{x_1 x_2}, \ldots, g^{x_{n-1}x_n}, g^{x_n x_1} \,|\, x_1, \ldots, x_n \in_R \mathbb{Z}_q\},$$

$$\mathsf{PDH}^{\$}_n = \{g^{x_1}, \ldots, g^{x_n}, g^{y_1}, \ldots, g^{y_n} \,|\, x_1, \ldots, x_n, y_1, \ldots, y_n \in_R \mathbb{Z}_q\}.$$

A (t, ϵ)-distinguisher against PDDH_n for $\mathbb{G}$ is a probabilistic Turing machine Δ with time-complexity t, which is able to distinguish these two distributions with an advantage $\mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(\Delta)$ greater than ϵ. The **advantage function** $\mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(t)$ for the group $\mathbb{G}$, is then defined as the maximum value of $\mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(\Delta)$ over all Δ with time-complexity at most t.

Lemma C.4.2 [Equivalence between PDDH_n and DDH] For any group $\mathbb{G}$ and any integer n, the *PDDH*$_n$ and the *DDH* problems are equivalent: for any time bound T,

$$\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(T) \leq \mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(T) \leq n\,\mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(T).$$

Proof: We omit the proof of this lemma in this version of the paper as it follows from a standard hybrid argument [Gol04, GM84] with $n+1$ hybrid experiments, in which the first i DDH values are replaced by random ones in the i-th hybrid experiment for $i \in \{0,\ldots,n\}$. In fact, a proof of this lemma was implicitly made in the proceedings version of the paper by Katz and Yung in Crypto 2003 [KY03] when showing an upper bound for the probability distance between the experiments Fake_n and Real. Moreover, in the full version of their paper, they provide an even tighter security reduction between these two problems. ▮

In our security analysis, we will need a challenger that outputs a new tuple either from $\mathsf{PDH}^{\star}_n$ or $\mathsf{PDH}^{\$}_n$, according to an input bit. That is, we have a fixed bit β, and for any new query S, $\mathsf{Chall}^{\beta}(S)$ outputs a new tuple from $\mathsf{PDH}^{\star}_n$ if $\beta = 0$, or from $\mathsf{PDH}^{\$}_n$ if $\beta = 1$. If the same S is queried again, then the same output tuple is returned. It is a well-known result that after q queries to the challenger, any adversary in time t cannot guess the bit β with advantage larger than $q \times \mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(t) \leq qn \times \mathsf{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t)$.

C.5 Proof of Theorem C.4.1

We proceed by defining several experiments (or *games*), the first one being the real-world experiment (in which the success of the adversary in outputting $b' = b$ — denoted by event S — is larger than $(1 + \mathsf{Adv}^{\mathsf{ake}}(\mathcal{A}))/2$ by definition), the last one being a *trivially secure* experiment in which the success of the adversary is straightforwardly 1/2.

Game $\mathbf{G}_9$: This is the real attack game, in the random-oracle and ideal-cipher models.

Game $\mathbf{G}_{10}$: We simulate the random oracles $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{A}uth$ in a classical way using the lists $\Lambda_{\mathcal{G}}$, $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{A}uth}$, with a random value for any new query, and we cancel executions (by halting the simulation and declaring the adversary successful) in which a collision occurs in the output of hash functions. The probability of such bad event is upper-bounded by the birthday paradox.

$$|\Pr[\mathsf{S}_{10}] - \Pr[\mathsf{S}_9]| \leq \frac{q^2_{\mathcal{G}}}{2^{\ell_{\mathcal{G}}}} + \frac{q^2_{\mathcal{H}}}{2^{\ell_{\mathcal{H}}}} + \frac{q^2_{\mathcal{A}uth}}{2^{\ell_{\mathcal{A}uth}}}.$$

Game $\mathbf{G}_{11}$: In this game, we start to control the simulation of the ideal cipher by maintaining a list Λ that keeps track of the previous queries-answers and that links each query to a specific user. Members of the list Λ are of the form $(type, S, i, \alpha, k, z, z^{\star})$, where $type \in \{\mathsf{enc}, \mathsf{dec}\}$. Such record means that $\mathcal{E}_k(z) = z^{\star}$, and $type$ indicates which kind of queries generated the record. The index i indicates which player is associated with the key k, while S indicates the session with which we are dealing. These values are both set to $\perp$ if k does not come from a $\mathcal{H}$ query of the form $(S, i, *)$ with $i \in \{1,\ldots,n\}$, and S of any form. The element α will be explained later.

- On encryption query $\mathcal{E}_k(z)$, we look for a record $(\cdot,\cdot,\cdot,\cdot,k,z,*)$ in Λ. If such a record exists, we return its last component. Otherwise, we choose uniformly at random $z^{\star} \in \mathbb{G}$, add $(\mathsf{enc}, \perp, \perp, \perp, k, z, z^{\star})$ to Λ, and return $z^{\star}$.
- On decryption query $\mathcal{D}_k(z^{\star})$, we look for a record $(\cdot,\cdot,\cdot,\cdot,k,*,z^{\star})$ in Λ. If such a record exists, we return its sixth component. Otherwise, we distinguish two sub-cases, by looking up in Λ_H if k has been returned to a hash query of the form $(S, i, *)$: if it the case,

we choose z at random in $\mathbb{G}^\star = \mathbb{G}\backslash\{0\}$ and update the list Λ with $(\mathsf{dec}, S, i, \perp, k, z, z^\star)$; otherwise, we choose z at random in $\mathbb{G}^\star$ and update the list Λ with $(\mathsf{dec}, \perp, \perp, \perp, k, z, z^\star)$. In both cases, the decryption query on $z^\star$ is answered with z.

Such a simulation is perfect, except for the following three points. First, collisions may appear that contradict the permutation property of the ideal-cipher: the probability can be upper-bounded by $(q_\mathcal{E} + q_\mathcal{D})^2/2|\mathbb{G}|$. Second, we avoided z being equal to 1 in the decryption queries. Finally, in the case of the decryption query simulation, one will cancel executions (by halting the simulation and declaring the the adversary successful) if the value k (involved in a decryption query) is output later by $\mathcal{H}$. Fortunately, this happens with probability at most $q_H/2^{\ell_\mathcal{H}}$ for each decryption query. Intuitively, as it will become clear in the next games, we indeed want to make sure that, for any k involved in a decryption query, if k comes from a $\mathcal{H}$ query, we know the corresponding pair (S, i). All being considered, such bad events are unlikely:

$$|\Pr[\mathsf{S}_{11}] - \Pr[\mathsf{S}_{10}]| \leq \frac{(q_\mathcal{E} + q_\mathcal{D})^2}{2|\mathbb{G}|} + \frac{q_\mathcal{D}}{|\mathbb{G}|} + \frac{q_\mathcal{H} q_\mathcal{D}}{2^{\ell_\mathcal{H}}}.$$

Game $\mathbf{G}_{12}$: In this game, we change the simulation of the decryption queries, and make use of our challenger to embed an instance of the PDH problem in the protocol simulation. In this game, we set $\beta = 0$, so that our challenger $\mathsf{Chall}^\beta(\cdot)$ output tuples $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$ according to the $\mathsf{PDH}^\star_n$ distribution. We use these $(2n)$-tuples to properly simulate the decryption queries.

More precisely, we issue a new tuple each time a new session S appears in a decryption query. But if several queries are asked with the same S, the challenger outputs the same tuple, so we will derive many related instances, granted the random self-reducibility. The latter tells us that, given one tuple outputted by the challenger, then for any randomly chosen $(\alpha_1, \ldots, \alpha_n)$, the tuple $(\zeta_1^{\alpha_1}, \ldots, \zeta_n^{\alpha_n}, \gamma_1^{\alpha_1\alpha_2}, \ldots, \gamma_n^{\alpha_n\alpha_1})$ has the same distribution as the original one.

We make use of this property as follows, by modifying the first sub-case previously considered for *new* decryption queries.

- On a new decryption query $\mathcal{D}_k(z^\star)$, such that $k = \mathcal{H}(S, i, *)$ was previously obtained from $\mathcal{H}$ for some valid index i, we query $\mathsf{Chall}^\beta(S)$ in order to get a tuple $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$. We then randomly choose $\alpha \in \mathbb{Z}_q^\star$, add $(\mathsf{dec}, S, i, \alpha, k, z = \zeta_i^\alpha, z^\star)$ to Λ, and return z.

Above, we have defined the list Λ whose elements are of the form $(type, S, i, \alpha, k, z, z^\star)$. The component '$\alpha$' now comes into play. This element is an exponent indicating how we applied the random self-reducibility of the PDDH problem, to the instance generated by the challenger upon the request S: $X = \zeta_i^\alpha$. Here, the element α can only be defined if S and i are known (in order to know which tuple, and which ζ_i, we are working with.) If α is unknown to the simulator, we set $\alpha = \perp$.

This change does not modify the view of the adversary, so: $\Pr[\mathsf{S}_{12}] = \Pr[\mathsf{S}_{11}]$.

Game $\mathbf{G}_{13}$: We are now ready to simulate the Send queries in a different way, but only in the second and third rounds: when the session S is defined, user i computes the symmetric keys as before $k_j = \mathcal{H}(S, j, pw)$, for all j. We thus know we are working with the tuple $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$.

In the second round, U_i randomly chooses a value $z_i^\star \in \mathbb{G}$ to be broadcasted, and asks $z_i = \mathcal{D}_{k_i}(z_i^\star)$, using the above simulation (which leads to add α_i to the list Λ, unless $z_i^\star$ already appeared as an encryption result. But the latter event cannot happen with probability greater than $q_\mathcal{E}/|\mathbb{G}|$.)

In the third round, U_i recovers $z_{i-1} = \mathcal{D}_{k_{i-1}}(z_{i-1}^\star)$ and $z_{i+1} = \mathcal{D}_{k_{i+1}}(z_{i+1}^\star)$. But then, two situations may appear:

- $z^\star_{i-1}$ and $z^\star_{i+1}$ have been simulated according to the above simulation of the second round, and then one gets α_{i-1} and α_{i+1} in the list Λ such that $z_{i-1} = \zeta_{i-1}^{\alpha_{i-1}}$ and $z_{i+1} = \zeta_{i+1}^{\alpha_{i+1}}$;
- one of the $z^\star_j$ has been previously answered by the encryption oracle in response to an attacker query $\mathcal{E}_k(z^\star)$, where $k = \mathcal{H}(S, j, pw)$ is the correct key for player U_j in session S. We denote such an event by $\mathsf{Encrypt}$. In such a case, we stop the simulation, letting the adversary win.

If everything runs smoothly, one gets

$$z_i = \zeta_i^{\alpha_i} \quad z_{i-1} = \zeta_{i-1}^{\alpha_{i-1}} \quad z_{i+1} = \zeta_{i+1}^{\alpha_{i+1}}.$$

One can then correctly compute

$$Z_i = \mathsf{CDH}(z_{i-1}, z_i) = \gamma_{i-1}^{\alpha_{i-1}\alpha_i} \quad Z_{i+1} = \mathsf{CDH}(z_i, z_{i+1}) = \gamma_i^{\alpha_i\alpha_{i+1}}.$$

One then broadcasts $X_i = Z_{i+1}/Z_i$. After this final round, everybody can compute the session key as before. The simulation is still perfect, unless the above bad events happen:

$$|\Pr[\mathsf{S}_{13}] - \Pr[\mathsf{S}_{12}]| \leq \frac{q_\mathcal{E} q_{\mathsf{passive}}}{|\mathbb{G}|} + \Pr[\mathsf{Encrypt}_{13}] \leq \frac{n q_\mathcal{E} q_{\mathsf{session}}}{|\mathbb{G}|} + \Pr[\mathsf{Encrypt}_{13}].$$

Game $\mathbf{G}_{14}$: Since it is clear that the security of the above scheme still relies on the DDH assumption, we now flip the bit β to 1, in order to receive tuples $(\zeta_1, \ldots, \zeta_n, \gamma_1, \ldots, \gamma_n)$ according to the $\mathsf{PDH}^{\$}_n$ distribution (in which the y_i's denote the values $\log_g \gamma_i$).

$$\begin{aligned} |\Pr[\mathsf{S}_{14}] - \Pr[\mathsf{S}_{13}]| &\leq q_{\mathsf{session}} \mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(t) \\ |\Pr[\mathsf{Encrypt}_{14}] - \Pr[\mathsf{Encrypt}_{13}]| &\leq q_{\mathsf{session}} \mathsf{Adv}^{\mathsf{pddh}_n}_{\mathbb{G}}(t). \end{aligned}$$

Game $\mathbf{G}_{15}$: In order to stop active attacks, where the adversary forges flows, we modify the computation of the key confirmations: we replace the function $\mathcal{A}uth$ by a private one $\mathcal{A}uth'$: $\mathcal{A}uth_i = \mathcal{A}uth'(S, \{z^\star_j, X_j\}_j, K_i, i)$, where

$$\begin{aligned} K_i &= Z_i^n X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2} = \gamma_{i-1}^{n\alpha_{i-1}\alpha_i} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2} \\ &= g^{n(\alpha_{i-1}\alpha_i y_{i-1})} X_i^{n-1} X_{i+1}^{n-2} \cdots X_{i+n-2}. \end{aligned}$$

Let us list all the information a (powerful) adversary may have, from all the X_j sent by U_j in the S-th session:

$$\log X_j = y_j(\alpha_j\alpha_{j+1}) - y_{j-1}(\alpha_{j-1}\alpha_j) = A_j y_j - A_{j-1} y_{j-1}.$$

As explained in [KY03], this does not leak any information about y_{i-1}, since the above system contains only $n-1$ independent equations with n unknowns. Any value for y_{n-1} is thus possible and would determine all the other values.

Therefore, after this modification, the probability for the adversary to see the difference between the current and the previous experiments is to query $\mathcal{A}uth(S, \{z^\star_j, X_j\}_j, K_i, i)$, which is upper-bounded by $q_{\mathcal{A}uth}/|\mathbb{G}|$.

$$|\Pr[\mathsf{S}_{15}] - \Pr[\mathsf{S}_{14}]| \leq \frac{q_{\mathcal{A}uth}}{|\mathbb{G}|} \qquad |\Pr[\mathsf{Encrypt}_{15}] - \Pr[\mathsf{Encrypt}_{14}]| \leq \frac{q_{\mathcal{A}uth}}{|\mathbb{G}|}.$$

Game $\mathbf{G}_{16}$: Finally, we now derive the session keys using a private random oracle $\mathcal{G}'$: $sk_i = \mathcal{G}'(S, \{z^\star_j, X_j, \mathcal{A}uth_j\}_j)$. As above, after the modification of the derivation of the session

key, the probability for the adversary to see the difference between the current and the previous experiments is to query $\mathcal{G}(S, \{z_j^\star, X_j, \mathcal{A}uth_j\}_j, K_i)$. Since the previous game, we know that inside each session, all the honest users have the same view, and thus theses queries are identical: the probability of such an event can also be upper-bounded by $q_\mathcal{G}/|\mathbb{G}|$, since no information has been leaked about K_i (except it does not correspond to the $\mathcal{A}uth$ queries asked above.)

$$\begin{aligned} \big|\Pr[\mathsf{S}_{16}] - \Pr[\mathsf{S}_{15}]\big| &\leq \frac{q_\mathcal{G}}{|\mathbb{G}| - q_{\mathcal{A}uth}} \leq \frac{2q_\mathcal{G}}{|\mathbb{G}|} \\ \big|\Pr[\mathsf{Encrypt}_{16}] - \Pr[\mathsf{Encrypt}_{15}]\big| &\leq \frac{q_\mathcal{G}}{|\mathbb{G}| - q_{\mathcal{A}uth}} \leq \frac{2q_\mathcal{G}}{|\mathbb{G}|}. \end{aligned}$$

Furthermore, because the private oracle $\mathcal{G}'$ is private to the simulator, it is clear that

$$\Pr[\mathsf{S}_{16}] = \frac{1}{2}.$$

Game $\mathbf{G}_{17}$: In order to conclude the proof, we need to upper-bound the event $\mathsf{Encrypt}_{16}$. One can note that the password pw is only used in the simulation of the second and third rounds, to compute z_i, z_{i-1} and z_{i+1} (using the elements ζ_i, ζ_{i-1} and ζ_{i+1}), but eventually, we output X_i only, which are computed from the γ_{i-1} and γ_i. The latter is totally independent of the former.

We can thus simplify the simulation of the second and third rounds: In the second round, U_i randomly chooses $z_i^\star \in \mathbb{G}$, and sends it (this is exactly as before.) However no decryption is needed. In the third round, U_i simply computes and sends $X_i = \gamma_i/\gamma_{i-1}$ (this is just to make sure that the product of the X_i is equal to 1, but we just need random elements satisfying this relation, since they do not appear anywhere else.) This is a perfect simulation, since one does not need anymore to compute K_i.

At this point, the password is never used, and can thus be chosen at the very end only, which makes clear that probability of the $\mathsf{Encrypt}$ event is less than the number of first flows manufactured by the adversary, divided by N. The latter part is upper-bounded by q_{active}:

$$\Pr[\mathsf{Encrypt}_{16}] = \Pr[\mathsf{Encrypt}_{17}] \leq q_{\mathsf{active}}/N.$$

In the above, we used the fact that collisions in the output of $\mathcal{H}$ have been eliminated in previous games.

Putting all equations together, one easily gets the announced bound.

C.6 Conclusion

We described a constant-round password-based key exchange protocol for group, derived from the Burmester-Desmedt scheme. The protocol is proven secure against dictionary attacks under the DDH assumption, in the ideal-cipher and random oracle models. It remains an open problem to find a scheme whose security depends on the number of active sessions rather than on the number of manufactured flows.

Acknowledgements

The first and fourth authors were supported in part by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure. The third author was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, Mathematical Information and Computing Sciences Division, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This document is report LBNL-59542. See `http://www-library.lbl.gov/disclaimer`.

Appendix D

A Scalable Password-based Group Key Exchange Protocol in the Standard Model

ASIACRYPT 2006
[AP06] with D. Pointcheval

Abstract : *This paper presents a secure constant-round password-based group key exchange protocol in the common reference string model. Our protocol is based on the group key exchange protocol by Burmester and Desmedt and on the 2-party password-based authenticated protocols by Gennaro and Lindell, and by Katz, Ostrovsky, and Yung. The proof of security is in the standard model and based on the notion of smooth projective hash functions. As a result, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and N-residuosity.*

D.1 Introduction

Key exchange is one of the most useful tools in public-key cryptography, allowing users to establish a common secret which they can then use in applications to achieve both privacy and authenticity. Among the examples of key exchange protocols, the most classical one is the Diffie-Hellman protocol [DH76]. Unfortunately, the latter only works between two players and does not provide any authentication of the players.

Group Key Exchange.

Group key exchange protocols are designed to provide a pool of players communicating over an open network with a shared secret key which may later be used to achieve cryptographic goals like multicast message confidentiality or multicast data integrity. Secure virtual conferences involving up to one hundred participants is an example.

Due to the usefulness of group key exchange protocols, several papers have attempted to extend the basic Diffie-Hellman protocol to the group setting. Nonetheless, most of these attempts were rather informal or quite inefficient in practice for large groups. To make the analyses of such protocols more formal, Bresson et al. [BCP01, BCPQ01] introduced a formal security model for

group key exchange protocols, in the same vein as [BR94a, BR95, BPR00]. Moreover, they also proposed new protocols, referred to as group Diffie-Hellman protocols, using a ring structure for the communication, in which each player has to wait for the message from his predecessor before producing his own. Unfortunately, the nature of their communication structure makes their protocols quite impractical for large groups since the number of rounds of communication is linear in the number of players.

A more efficient and practical approach to the group key exchange problem is the one proposed by Burmester and Desmedt [BD94, BD05], in which they provide a *constant-round* Diffie-Hellman variant. Their protocol is both scalable and efficient, even for large groups, since it only requires 2 rounds of broadcasts. Thus, with reasonable time-out values, one could always quickly decide whether or not a protocol has been successfully executed. Furthermore, their protocol has also been formally analyzed, in the above security framework [KY03].

Password-Based Authenticated Key Exchange.

The most classical way to add authentication to key exchange protocols is to sign critical message flows. In fact, as shown by Katz and Yung [KY03] in the context of group key exchange protocols, this technique can be made quite general and efficient, converting any scheme that is secure against passive adversaries into one that is secure against active ones. Unfortunately, such techniques require the use of complex infrastructures to handle public keys and certificates. One way to avoid such infrastructures is to use passwords for authentication. In the latter case, the pool of players who wants to agree on a common secret key only needs to share a low-entropy password —a 4-digit pin-code, for example— against which an exhaustive search is quite easy to perform. In password-based protocols, it is clear that an outsider attacker can always guess a password and attempt to run the protocol. In case of failure, he can try again with a different guess. After each failure, the adversary can erase one password. Such an attack, known as "on-line exhaustive search" cannot be avoided, but the damage it may cause can be mitigated by other means such as limiting the number of failed login attempts. A more dangerous threat is the "off-line exhaustive search", also known as "dictionary attack". It would mean that after one failure, or even after a simple eavesdropping, the adversary can significantly reduce the number of password candidates.

In the two-party case, perhaps the most well known Diffie-Hellman variant is the encrypted key exchange protocol by Bellovin and Merritt [BM92]. However, its security analyses [BPR00, BMP00, BCP03, BCP04] require ideal models, such as the random oracle model [BR93] or the ideal cipher model. The first practical password-based key exchange protocol, without random oracles, was proposed by Katz et al. [KOY01] in the common reference string model and it is based on the Cramer-Shoup cryptosystem [CS98]. Their work was later extended by Gennaro and Lindell [GL03] using the more general smooth projective hash function primitive [CS98, CS02, CS03].

In the group key exchange case, very few protocols have been proposed with password authentication. In [BCP02b, BCP07], Bresson et al. showed how to adapt their group Diffie-Hellman protocols to the password-based scenario. However, as the original protocols on which they are based, their security analyses require ideal models and the total number of rounds is linear in the number of players, making their schemes impractical for large groups. More recently, several constant-round password-based group key exchange protocols have been proposed in the literature by Abdalla et al. [ABCP06], by Dutta and Barua [DB06], and by Kim, Lee, and Lee [KLL04]. All of these constructions are based on the Burmester and Desmedt protocol [BD94, BD05] and are quite efficient, but their security analyses usually require the random oracle and/or the ideal cipher models. [1] Independently and concurrently to our work,

[1] In fact, in [ABCP06], Abdalla et al. showed that the protocols by Dutta and Barua [DB06] and by Kim, Lee,

a new constant-round password-based group key exchange protocol has been proposed by Bohli et al. [BGS06]. Their protocol is more efficient than ours and also enjoys a security proof in the standard model.

Contributions.

In this paper, we propose the first password-based authenticated group key exchange protocol in the standard model. To achieve this goal, we extend the Gennaro-Lindell framework [GL03] to the group setting, using ideas similar to those used in the Burmester-Desmedt protocol [BD94, BD05]. In doing so, we take advantage of the smooth projective hash function primitive [CS02] to avoid the use of ideal models. Our protocol has several advantages. First, it is efficient both in terms of communication, only requiring 5 rounds, and in terms of computation, with a per-user computational load that is linear in the size of the group. Second, like the Burmester-Desmedt protocol, our protocol is also contributory since each member contributes equally to the generation of the common session key. Such property, as pointed out by Steiner, Tsudik and Waidner [STW00], may be essential for certain distributed applications. Finally, as in the Gennaro-Lindell framework [GL03], our protocol works in the common reference string model and is quite general, being built in a modular way from four cryptographic primitives: a LPKE-IND-CCA-secure labeled encryption scheme, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, it can be instantiated under various computational assumptions, such as decisional Diffie-Hellman, quadratic residuosity, and N-residuosity (see [GL03]). In particular, the Diffie-Hellman variant (based on the Cramer-Shoup cryptosystem [CS98]) can be seen as a generalization of the KOY protocol [KOY01] to the group setting.

D.2 Security Model

The security model for password-based group key exchange protocols that we present here is the one by Bresson et al. [BCP07], which is based on the model by Bellare et al. [BPR00] for 2-party password-based key exchange protocols.

Protocol participants.

Let $\mathcal{U}$ denote the set of potential participants in a password-based group key exchange protocol. Each participant $U \in \mathcal{U}$ may belong to several subgroups $\mathcal{G} \subseteq \mathcal{U}$, each of which has a unique password $\mathsf{pw}_{\mathcal{G}}$ associated to it. The password $\mathsf{pw}_{\mathcal{G}}$ of a subgroup $\mathcal{G}$ is known to all the users $U_i \in \mathcal{G}$.

Protocol execution.

The interaction between an adversary $\mathcal{A}$ and the protocol participants only occurs via oracle queries, which model the adversary capabilities in a real attack. During the execution of the protocol, the adversary may create several instances of a participant and several instances of the same participant may be active at any given time. Let $U^{\langle i \rangle}$ denote the instance i of a participant U and let b be a bit chosen uniformly at random. The query types available to the adversary are as follows:

- $\mathit{Execute}(U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle})$: This query models passive attacks in which the attacker eavesdrops on honest executions among the participant instances $U_1^{\langle i_1 \rangle}, \ldots, U_n^{\langle i_n \rangle}$. It returns the messages that were exchanged during an honest execution of the protocol.

and Lee are insecure by presenting concrete attacks against these schemes.

- *Send*($U^{\langle i \rangle}, m$): This query models an active attack, in which the adversary may tamper with the message being sent over the public channel. It returns the message that the participant instance $U^{\langle i \rangle}$ would generate upon receipt of message m.
- *Reveal*($U^{\langle i \rangle}$): This query models the misuse of session keys by a user. It returns the session key held by the instance $U^{\langle i \rangle}$.
- *Test*($U^{\langle i \rangle}$): This query tries to capture the adversary's ability to tell apart a real session key from a random one. It returns the session key for instance $U^{\langle i \rangle}$ if $b = 1$ or a random key of the same size if $b = 0$.

Partnering.

Following [KY03], we define the notion of partnering via session and partner identifiers. Let the session identifier sid^i of a participant instance $U^{\langle i \rangle}$ be a function of all the messages sent and received by $U^{\langle i \rangle}$ as specified by the group key exchange protocol. Let the partner identifier pid^i of a participant instance $U^{\langle i \rangle}$ is the set of all participants with whom $U^{\langle i \rangle}$ wishes to establish a common secret key. Two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are said to be partnered if and only if $\mathsf{pid}_1^{i_1} = \mathsf{pid}_2^{i_2}$ and $\mathsf{sid}_1^{i_1} = \mathsf{sid}_2^{i_2}$.

Freshness.

Differently from [KY03], our definition of freshness does not take into account forward security as the latter is out of the scope of the present paper. Let acc^i be true if an instance $U^{\langle i \rangle}$ goes into an accept state after receiving the last expected protocol message and false otherwise. We say that an instance $U^{\langle i \rangle}$ is fresh if $\mathsf{acc}^i = \mathsf{true}$ and no *Reveal* has been asked to $U^{\langle i \rangle}$ or to any of its partners.

Correctness.

For a protocol to be correct, it should always be the case that, whenever two instances $U_1^{\langle i_1 \rangle}$ and $U_2^{\langle i_2 \rangle}$ are partnered and have accepted, both instances should hold the same non-null session key.

Indistinguishability.

Consider an execution of the group key exchange protocol P by an adversary $\mathcal{A}$, in which the latter is given access to the *Reveal*, *Execute*, *Send*, and *Test* oracles and asks a single *Test* query to a *fresh* instance, and outputs a guess bit b'. Let Succ denote the event b' correctly matches the value of the hidden bit b used by the *Test* oracle. The AKE-IND advantage of an adversary $\mathcal{A}$ in violating the indistinguishability of the protocol P and the advantage function of the protocol P, when passwords are drawn from a dictionary $\mathcal{D}$, are respectively $\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(\mathcal{A}) = 2 \cdot \Pr[\,\text{Succ}\,] - 1$ and $\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(t, R) = \max_{\mathcal{A}}\{\mathbf{Adv}^{\text{ake-ind}}_{P,\mathcal{D}}(\mathcal{A})\}$, where maximum is over all $\mathcal{A}$ with time-complexity at most t and using resources at most R (such as the number of queries to its oracles). The definition of time-complexity that we use henceforth is the usual one, which includes the maximum of all execution times in the experiments defining the security plus the code size.

We say that a password-based group key exchange protocol P is secure if the advantage of any polynomial-time adversary is only negligibly larger than $O(q/|\mathcal{D}|)$, where q is number of different protocol instances to which the adversary has asked *Send* queries. Given that the dictionary size can be quite small in practice, the hidden constant in the big-O notation should be as small as possible (preferably 1) for a higher level of security.

D.3 Building blocks

D.3.1 Universal Hash Function Families

One of the tools used in our protocol is a family of universal hash functions. A family $\mathcal{UH}$ of universal hash function is a map $\mathbf{K} \times \mathbf{G} \mapsto \mathbf{R}$, where $\mathbf{K}$ is the key or seed space, $\mathbf{G}$ is the domain of the hash function, and $\mathbf{R}$ is the range. For each seed or key $k \in \mathbf{K}$, we can define a particular instance $\mathsf{UH}_k : \mathbf{G} \mapsto \mathbf{R}$ of the family by fixing the key being used in the computation of the function. For simplicity, we sometimes omit the seed k from the notation when referring to a particular instance of the family. Let UH_k be a universal hash function chosen at random from a family $\mathcal{UH}$. One of the properties of universal hash function families in which we are interested is the one that says that, if an element g is chosen uniformly at random from $\mathbf{G}$, then the output distribution of $\mathsf{UH}_k(g)$ is statistically close to uniform in $\mathbf{R}$ [HILL99].

D.3.2 Signatures

The signature scheme used in our protocol is the standard one introduced by Goldwasser, Micali, and Rivest [GMR88]. A standard signature scheme $\mathcal{SIG} = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{Ver})$ is composed of three algorithms. The key generation algorithm SKG takes as input 1^k, where k is a security parameter, and returns a pair (sk, vk) containing the secret signing key and the public verification key. The signing algorithm Sign takes as input the secret key sk and a message m and returns a signature σ for that message. The verification algorithm Ver on input (vk, m, σ) returns 1 if σ is a valid signature for the message m with respect to the verification key vk.

The security notion for signature schemes needed in our proofs is strong existential unforgeability under chosen-message attacks [GMR88]. More precisely, let (sk, vk) be a pair of secret and public keys for a signature scheme $\mathcal{SIG}$, let $\textsc{Sign}(\cdot)$ be a signing oracle which returns $\sigma = \mathsf{Sign}(sk, m)$ on input m, and let $\mathcal{F}$ be an adversary. Then, consider the experiment in which the adversary $\mathcal{F}$, who is given access to the public key vk and to the signing oracle $\textsc{Sign}(\cdot)$, outputs a pair (m, σ). Let $\{(m_i, \sigma_i)\}$ denote the set of queries made to the signing oracle with the respective responses and let $\textsc{Succ}$ denote the event in which $\mathsf{Ver}(vk, m', \sigma') = 1$ and that $(m', \sigma') \notin \{(m_i, \sigma_i)\}$. The SIG-SUF-CMA-*advantage* of an adversary $\mathcal{F}$ in violating the chosen message security of the signature scheme $\mathcal{SIG}$ is defined as $\mathbf{Adv}^{\text{sig-suf-cma}}_{\mathcal{SIG},\mathcal{F}}(k) = \Pr[\textsc{Succ}]$. A signature scheme $\mathcal{SIG}$ is said to be SIG-SUF-CMA-*secure* if this advantage is a negligible function in k for all polynomial time adversaries (PTAs) $\mathcal{F}$ asking a polynomial number of queries to their signing oracle.

D.3.3 Labeled Encryption

The notion of labeled encryption, first formalized in the ISO 18033-2 standard [Sho04], is a variation of the usual encryption notion that takes into account the presence of labels in the encryption and decryption algorithms. More precisely, in a labeled encryption scheme, both the encryption and decryption algorithms have an additional input parameter, referred to as a label, and the decryption algorithm should only correctly decrypt a ciphertext if its input label matches the label used to create that ciphertext.

Formally, a labeled encryption scheme $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms. Via $(pk, sk) \xleftarrow{\$} \mathsf{LKG}(1^k)$, where $k \in \mathbb{N}$ is a security parameter, the randomized key-generation algorithm produces the public and secret keys of the scheme. Via $c \xleftarrow{\$} \mathsf{Enc}(pk, l, m; r)$, the randomized encryption algorithm produces a ciphertext c for a label l and message m using r as the randomness. Via $m \leftarrow \mathsf{Dec}(sk, l, c)$, the decryption algorithm decrypts the ciphertext c using l as the label to get back a message m.

The security notion for labeled encryption is similar to that of standard encryption schemes. The main difference is that, whenever the adversary wishes to ask a query to his Left-or-Right encryption oracle, in addition to providing a pair of messages (m_0, m_1), he also has to provide a target label l in order to obtain the challenge ciphertext c. Moreover, when chosen-ciphertext security (LPKE-IND-CCA) is concerned, the adversary is also allowed to query his decryption oracle on any pair (l, c) as long as the ciphertext c does not match the output of a query to his Left-or-Right encryption oracle whose input includes the label l. Formally, let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a labeled encryption scheme. To any bit $b \in \{0,1\}$ and any adversary $\mathcal{D}$, we associate the experiment:

Experiment $\mathbf{Exp}^{\text{lpke-ind-cca-}b}_{\mathcal{LPKE},\mathcal{D}}(k)$	Oracle $\textsc{Enc}(l, m_0, m_1)$
$(pk, sk) \xleftarrow{\$} \mathsf{LKG}(1^k)$	$c \xleftarrow{\$} \mathsf{Enc}(pk, l, m_b)$
$\boldsymbol{EncList} \leftarrow \emptyset$; $\boldsymbol{DecList} \leftarrow \emptyset$	$\boldsymbol{EncList} \leftarrow \boldsymbol{EncList} \cup \{(l, c)\}$
$b' \xleftarrow{\$} \mathcal{D}^{\textsc{Enc}(\cdot,\cdot,\cdot),\textsc{Dec}(\cdot,\cdot)}(pk)$	return c
if $(\boldsymbol{EncList} \cap \boldsymbol{DecList} = \emptyset)$	Oracle $\textsc{Dec}(l, c)$
then return b' else return 0	$\boldsymbol{DecList} \leftarrow \boldsymbol{DecList} \cup \{(l, c)\}$
	return $\mathsf{Dec}(sk, l, c)$

The LPKE-IND-CCA-*advantage* of an adversary $\mathcal{D}$ in violating the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$ is defined as

$$\mathbf{Adv}^{\text{lpke-ind-cca}}_{\mathcal{LPKE},\mathcal{D}}(k) = \Pr\left[\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k) = 1\right].$$

A LPKE scheme $\mathcal{LPKE}$ is said to be LPKE-IND-CCA-*secure* if this advantage is a negligible function in k for all PTAs $\mathcal{D}$. As shown by Bellare et al. in the case of standard encryption schemes [BBM00], one can easily show that the Left-or-Right security notion for labeled encryption follows from the more standard Find-Then-Guess security notion (in which the adversary is only allowed a single query to his challenging encryption oracle).

D.3.4 Smooth Projective Hash Functions

The notion of projective hash function families was first introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext secure encryption schemes. Later, Gennaro and Lindell [GL03] showed how to use such families to build secure password-based authenticated key exchange protocols. One of the properties that makes these functions particularly interesting is that, for certain points of their domain, their values can be computed by using either a *secret* hashing key or a *public* projective key. While the computation using *secret* hashing key works for all the points in the domain of the hash function, the computation using a public *projective* key only works for a specified subset of the domain. A projective hash function family is said to be smooth if the value of the function on inputs that are outside the particular subset of the domain are independent of the projective key. In [GL03], the notion of smooth hash functions was presented in the context of families of hard (partitioned) subset membership problems. Here we follow the same approach.

Hard partitioned subset membership problems. Let $k \in \mathbb{N}$ be a security parameter. In a family of hard (partitioned) subset membership problem, we first specify two sets $\mathbf{X}(k)$ and $\mathbf{L}(k)$ in $\{0,1\}^{\mathbf{poly}(k)}$ such that $\mathbf{L}(k) \subseteq \mathbf{X}(k)$ as well as two distributions $D(\mathbf{L}(k))$ and $D(\mathbf{X}(k) \setminus \mathbf{L}(k))$ over $\mathbf{L}(k)$ and $\mathbf{X}(k) \setminus \mathbf{L}(k))$ respectively. Next, we specify a witness set $\mathbf{W}(k) \subseteq \{0,1\}^{\mathbf{poly}(k)}$ and a NP-relation $\mathbf{R}(k) \subseteq \mathbf{X}(k) \times \mathbf{W}(k)$ such that $x \in \mathbf{L}(k)$ if and only if there exists a witness $w \in \mathbf{W}(k)$ such that $(x, w) \in \mathbf{R}(k)$. Then, we say that a family of subset membership problems is hard if $(\mathbf{X}(k), \mathbf{L}(k), D(\mathbf{L}(k)), D(\mathbf{X}(k) \setminus \mathbf{L}(k)), \mathbf{W}(k), \mathbf{R}(k))$ instances can be efficiently generated, that a member element $x \in \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{L}(k))$ along with a

witness $w \in \mathbf{W}(k)$ to the fact that $(x, w) \in \mathbf{R}(k)$, that non-member elements $x \in \mathbf{X}(k) \setminus \mathbf{L}(k)$ can be efficiently sampled according to $D(\mathbf{X}(k) \setminus \mathbf{L}(k))$, and that the distributions of member and non-member elements cannot be efficiently distinguished. The definition of hard *partitioned* subset membership problem is an extension of the one given above in which the set $\mathbf{X}(k)$ is partitioned in disjoint subsets $\mathbf{X}(k, i)$ for some index $i \in \{1, \ldots, l\}$ and for which for all i it remains hard to distinguish an element $x \in \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{L}(k, i))$ from an element $x \in \mathbf{X}(k, i) \setminus \mathbf{L}(k, i)$ chosen according to a distribution $D(\mathbf{X}(k, i) \setminus \mathbf{L}(k, i))$.

Hard partitioned subset membership problems from labeled encryption. The families of hard partitioned subset membership problems in which we are interested are those based on LPKE-IND-CCA-secure labeled encryption schemes. More precisely, let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a LPKE-IND-CCA-secure labeled encryption scheme and let pk be a public key outputted by the LKG algorithm for a given security parameter k. Let $\mathsf{Enc}(pk)$ denote an efficiently recognizable superset of the space of all ciphertexts that may be outputted by the encryption algorithm Enc when the public key is pk and let $\mathbf{L}$ and $\mathbf{M}$ denote efficiently recognizable supersets of the label and message spaces. Using these sets, we can define a family of hard partitioned subset membership problems as follows. First, we define the sets $\mathbf{X}$ and $\mathbf{L}$ for the family of hard subset membership problems as $\mathbf{X}(pk) = \mathsf{Enc}(pk) \times \mathbf{L} \times \mathbf{M}$ and $\mathbf{L}(pk) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \mathsf{Enc}(pk, l, m; r)\}$. Next, we define the partitioning of the sets $\mathbf{X}$ and $\mathbf{L}$ with respect to the message and label used in the encryption as $\mathbf{X}(pk, l, m) = \mathsf{Enc}(pk) \times l \times m$ and $\mathbf{L}(pk, l, m) = \{(c, l, m) \mid \exists r \text{ s.t. } c = \mathsf{Enc}(pk, l, m; r)\}$. The distribution $D(\mathbf{L}(pk, l, m))$ can then be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\mathsf{Enc}(pk, l, m; r), l, m)$ with r as a witness. Likewise, the distribution $D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m))$ can be defined by choosing a random $r \in \mathbf{R}$ and outputting the triple $(\mathsf{Enc}(pk, l, m'; r), l, m)$, where m' is a dummy message different from m but of the same length. Finally, we define the witness set $\mathbf{W}(pk)$ to be r and the NP-relation $\mathbf{R}(pk)$ in a natural way. It is easy to see that the hardness of distinguishing non-members from members follows from the LPKE-IND-CCA security of the labeled encryption scheme.

Smooth projective hash functions. Let $\mathcal{HLPKE}(pk) = (\mathbf{X}(pk), \mathbf{L}(pk), D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)), D(\mathbf{L}(pk, l, m)), \mathbf{W}(pk), \mathbf{R}(pk))$ be a family of hard (partitioned) subset membership problems based on a LPKE-IND-CCA-secure labeled encryption scheme $\mathcal{LPKE}$ with security parameter k. A family of smooth projective hash functions $\mathcal{HASH}(pk) = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ associated with $\mathcal{HLPKE}$ consists of four algorithms. Via $hk \xleftarrow{\$} \mathsf{HashKG}(pk)$, the randomized key-generation algorithm produces hash keys $hk \in \mathbf{HK}(pk)$, where $k \in \mathbb{N}$ is a security parameter and pk is the public key of a labeled encryption scheme $\mathcal{LPKE}$. Via $phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c)$, the randomized key projection algorithm produces projected hash keys $phk \in \mathbf{PHK}(pk)$ for a hash key hk with respect to label l and ciphertext c. Via $g \leftarrow \mathsf{Hash}(hk, c, l, m)$, the hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of (c, l, m) using the hash key hk. Via $g \leftarrow \mathsf{ProjHash}(phk, c, l, m; r)$, the projected hashing algorithm computes the hash value $g \in \mathbf{G}(pk)$ of (c, l, m) using the projected hash key phk and a witness r to the fact that c is a valid encryption of message m with respect to the public-key pk and label l.

Properties. The properties of smooth projective hash functions in which we are interested are correctness, smoothness, and pseudorandomness.

Correctness.

Let $\mathcal{LPKE}$ be a labeled encryption scheme and let pk be a public key outputted by the LKG algorithm for a given security parameter k. Let $c = \mathsf{Enc}(pk, l, m; r)$ be the ciphertext for a message m with respect to public key pk and label l computed using r as the randomness. Then, for any hash key $hk \in \mathbf{HK}(pk)$ and projected hash key $phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c)$, the values

$\mathsf{Hash}(hk, c, l, m)$ and $\mathsf{ProjHash}(phk, c, l, m, r)$ are the same.

Smoothness.

Let $hk \in \mathbf{HK}(pk)$ be a hash key and let $phk \in \mathbf{PHK}(pk)$ be a projected hash key for hk with respect to l and c. Then, for every triple (c, l, m) for which c is *not* a valid encryption of message m with respect to the public-key pk and label l (i.e., $(c, l, m) \in \mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)$), the hash value $g = \mathsf{Hash}(hk, c, l, m)$ is statistically close to uniform in $\mathbf{G}$ and independent of the values (phk, c, l, m).

Pseudorandomness.

Let $\mathcal{LPKE}$ be a LPKE-IND-CCA-secure labeled encryption scheme, let pk be a public key outputted by the LKG algorithm for a given security parameter k, and let $(l, m) \in \mathbf{L} \times \mathbf{M}$ be a message-label pair. Then, for uniformly chosen hash key $hk \in \mathbf{HK}(pk)$ and randomness $r \in \mathbf{R}(pk)$, the distributions $\{c = \mathsf{Enc}(pk, l, m; r), l, m, phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c), g \leftarrow \mathsf{Hash}(hk, c, l, m)\}$ and $\{c = \mathsf{Enc}(pk, l, m; r), l, m, phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c), g \xleftarrow{\$} \mathbf{G}\}$ are computationally indistinguishable.

More formally, let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a LPKE-IND-CCA-secure labeled encryption scheme, pk be a public key outputted by the LKG algorithm for a given security parameter k, and let $\mathcal{HLPKE} = (\mathbf{X}(pk), \mathbf{L}(pk), D(\mathbf{X}(pk, l, m) \setminus \mathbf{L}(pk, l, m)), D(\mathbf{L}(pk, l, m)), \mathbf{W}(pk), \mathbf{R}(pk))$ be a family of hard (partitioned) subset membership problems based on $\mathcal{LPKE}$. To any adversary $\mathcal{D}$, consider the experiments $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$, defined as follows.

$\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$ $(pk, sk) \xleftarrow{\$} \mathsf{LKG}(1^k)$ $\boldsymbol{EncList} \leftarrow \emptyset$ $b \leftarrow \mathcal{D}^{\text{ENC}(\cdot,\cdot),\text{HASH}(\cdot,\cdot,\cdot)}(pk)$ return b	Oracle $\text{ENC}(l, m)$ $c \xleftarrow{\$} \mathsf{Enc}(pk, l, m)$ $\boldsymbol{EncList} \leftarrow \boldsymbol{EncList} \cup \{(l, m, c)\}$ return c	Oracle $\text{HASH}(l, m, c)$ if $(l, m, c) \notin \boldsymbol{EncList}$ then return $\perp$ $hk \xleftarrow{\$} \mathsf{HashKG}(pk)$ $phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c)$ $\boxed{g \leftarrow \mathsf{Hash}(hk, l, m, c)}$ return (phk, g)
$\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$ $(pk, sk) \xleftarrow{\$} \mathsf{LKG}(1^k)$ $\boldsymbol{EncList} \leftarrow \emptyset$ $b \leftarrow \mathcal{D}^{\text{ENC}(\cdot,\cdot),\text{HASH}(\cdot,\cdot,\cdot)}(pk)$ return b	Oracle $\text{ENC}(l, m)$ $c \xleftarrow{\$} \mathsf{Enc}(pk, l, m)$ $\boldsymbol{EncList} \leftarrow \boldsymbol{EncList} \cup \{(l, m, c)\}$ return c	Oracle $\text{HASH}(l, m, c)$ if $(l, m, c) \notin \boldsymbol{EncList}$ then return $\perp$ $hk \xleftarrow{\$} \mathsf{HashKG}(pk)$ $phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c)$ $\boxed{g \xleftarrow{\$} \mathbf{G}}$ return (phk, g)

Then, for every (non-uniform) PTAs $\mathcal{D}$, the advantage $\mathbf{Adv}^{\text{h-prf}}_{\mathcal{HASH},\mathcal{D}}(k) = \Pr\left[\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{LPKE},\mathcal{D}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{LPKE},\mathcal{D}}(k) = 1\right]$ is a negligible function in k.

EXAMPLES. To provide the reader with an idea of how efficient smooth projective hash functions are, we recall here the example given in [GL03] based on the Cramer-Shoup encryption scheme [CS98].

The labeled version of the Cramer-Shoup scheme works as follows. Let G be a cyclic group of prime order q where q is large. The key generation algorithm chooses two additional random generators g_1, g_2 in G, a collision-resistant hash function H, and random values z, $\tilde{z}_1$, $\tilde{z}_2$, $\hat{z}_1$, $\hat{z}_2$ in Z_q with $z \neq 0$. The secret key is set to $(z, \tilde{z}_1, \tilde{z}_2, \hat{z}_1, \hat{z}_2)$ and the public key is defined to be $(h, \tilde{h}, \hat{h}, g_1, g_2, H)$, where $h = g_1^z$, $\tilde{h} = g_1^{\tilde{z}_1} g_2^{\tilde{z}_2}$, and $\hat{h} = g_1^{\hat{z}_1} g_2^{\hat{z}_2}$. To encrypt a message $m \in G$

with respect to label l, the sender chooses $r \in Z_q$, and computes $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r \cdot m$, $\theta = H(l, u_1, u_2, e)$ and $v = (\tilde{h}\hat{h}^\theta)^r$. The ciphertext is $c = (u_1, u_2, e, v)$. To decrypt a ciphertext $c = (u_1, u_2, e, v)$ with respect to label l, the receiver computes $\theta = H(l, u_1, u_2, e)$ and tests if v equals $u_1^{\tilde{z}_1+\theta\hat{z}_1} u_2^{\tilde{z}_2+\theta\hat{z}_2}$. If equality does not hold, it outputs $\perp$; otherwise, it outputs $m = eu_1^{-z}$.

The smooth projective hashing for the labeled Cramer-Shoup encryption scheme is then defined as follows. The hash key generation algorithm HashKG simply sets the key hk to be the tuple (a_1, a_2, a_3, a_4) where each a_i is a random value in Z_q. The key projection function ProjKG, on input (hk, l, c), first computes $\theta = H(l, u_1, u_2, e)$ and outputs $phk = g_1^{a_1} g_2^{a_2} h^{a_3} (\tilde{h}\hat{h}^\theta)^{a_4}$. The hash function Hash on input (hk, c, l, m) outputs $u_1^{a_1} u_2^{a_2} (e/m)^{a_3} v^{a_4}$. The projective hash function ProjHash on input (phk, c, l, m, r) simply outputs phk^r.

D.4 A scalable password-based group key exchange protocol

In this section, we finally present our password-based group key exchange protocol. Our protocol is an extension of the Gennaro-Lindell password-based key exchange protocol [GL03] to the group setting and uses ideas similar to those used in the Burmester-Desmedt group key exchange protocol [BD05]. The Gennaro-Lindell protocol itself is an abstraction of the password-based key exchange protocol of Katz, Ostrovsky, and Yung [KOY01, KOY02]. Like the Gennaro-Lindell protocol, our protocol is built in a modular way from four cryptographic primitives: a LPKE-IND-CCA-secure labeled encryption scheme, a signature scheme, a family of smooth projective hash functions, and a family of universal hash functions. Thus, our protocol enjoys efficient instantiations based on the decisional Diffie-Hellman, quadratic residuosity, and N-residuosity assumptions (see [GL03]). Like the Burmester-Desmedt group key exchange protocol, our protocol only requires a constant number of rounds and low per-user computation.

As done in the Gennaro-Lindell protocol, we also assume the existence of a mechanism to allow parties involved in the protocol to differentiate between concurrent executions as well as identify the other parties with which they are interacting. As in their case, this requirement is only needed for the correct operation of the protocol. No security requirement is imposed on this mechanism.

D.4.1 Protocol Description

Overview. As in the Burmester-Desmedt protocol, our protocol assumes a ring structure for the users so that we can refer to the predecessor and successor of a user. Moreover, we associate each user with an index i between 1 and n, where n is the size of the group. After deciding on the order of the users, our protocol works as follows. First, each user in the group executes two correlated instances of the Gennaro-Lindell protocol, one with his predecessor and one with his successor so each user can authenticate his neighbors (this accounts for the first 3 rounds of the protocol). However, instead of generating a single session key in each of these instances, we modify the original Gennaro-Lindell protocol so that two independent session keys are generated in each session (this requires an extra hash key and an extra projection key per user). We then use the first one of these as a test key to authenticate the neighbor with whom that key is shared and we use the other one to help in the computation of the group session key, which is defined as the product of these latter keys. To do so, we add one more round of communication like in the Burmester-Desmedt protocol, so that each user computes and broadcasts the ratio of the session keys that he shares with his predecessor and successor. After this round, each user is capable of computing the group session key. However, to ensure that all users agree on the same key, a final round of signatures is added to the protocol to make sure that all users compute the group session key based on the same transcript. The key used to verify the signature of a user

is the same one transmitted by that user in the first round of the Gennaro-Lindell protocol.

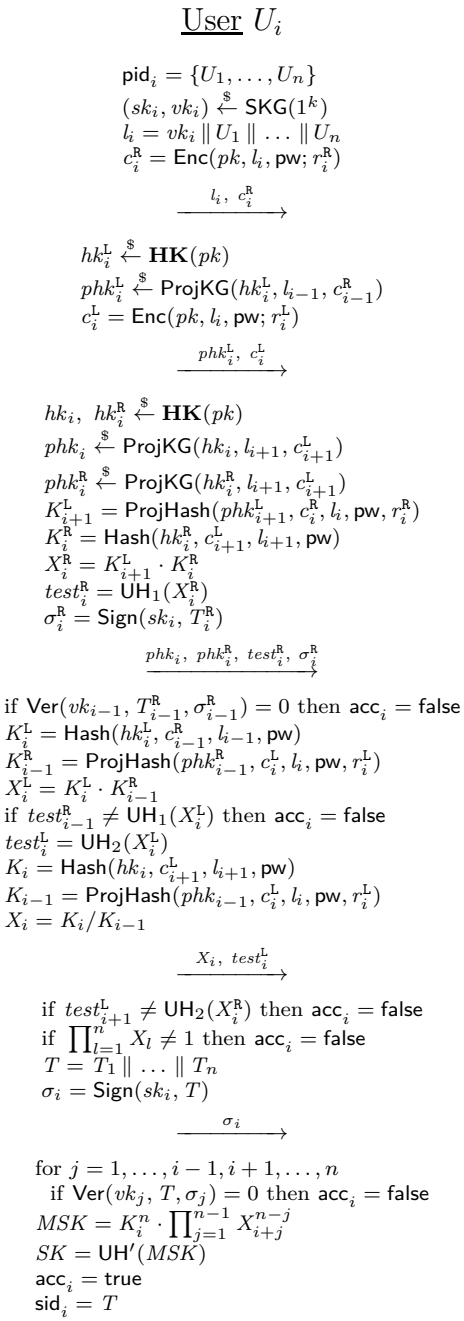


Figure D.1: An honest execution of the password-authenticated group key exchange protocol by player U_i in a group $\{U_1, \ldots, U_n\}$, where $T_i^{\mathtt{R}} = U_i \,\|\, U_{i+1} \,\|\, c_i^{\mathtt{R}} \,\|\, c_{i+1}^{\mathtt{L}} \,\|\, phk_i \,\|\, phk_i^{\mathtt{R}} \,\|\, phk_{i+1}^{\mathtt{L}} \,\|\, test_i^{\mathtt{R}}$ and $T_i = vk_i \,\|\, U_i \,\|\, c_i \,\|\, phk_i \,\|\, phk_i^{\mathtt{L}} \,\|\, phk_i^{\mathtt{R}} \,\|\, X_i \,\|\, X_i^{\mathtt{L}}$ for $i = 1, \ldots, n$.

For a pictorial description of our protocol, please refer to Figure D.1. The formal description follows.

Description. Let $\mathcal{LPKE} = (\mathsf{LKG}, \mathsf{Enc}, \mathsf{Dec})$ be a labeled encryption scheme, let $\mathcal{SIG} = (\mathsf{SKG}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme, and let $\mathcal{HASH}(pk) = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ be a family smooth projective hash functions based on $\mathcal{LPKE}$. Let $\mathsf{UH} : \mathbf{G} \mapsto \{0,1\}^{2l}$ and $\mathsf{UH}' : \mathbf{G} \mapsto \{0,1\}^{l}$ be two universal hash functions chosen uniformly at random from the

families $\mathcal{UH}$ and $\mathcal{UH}'$ and let $\mathsf{UH}_1(g)$ and $\mathsf{UH}_2(g)$ refer to the first and second halves of $\mathsf{UH}(g)$. Let $U_1, \ldots, U_n$ be the users wishing to establish a common secret key and let pw be their joint password chosen uniformly at random from a dictionary **Dict** of size N. We assume pw either lies in the message space $\mathbf{M}$ of $\mathcal{LPKE}$ or can be easily mapped to it. Our protocol has a total of five rounds of communication and works as follows.

Initialization.

A trusted server runs the key generation algorithm LKG on input 1^k, where $k \in \mathbb{N}$ is a security parameter, to obtain a pair (pk, sk) of secret and public keys and publishes the public key pk along with randomly selected universal hash function UH and UH' from the families $\mathcal{UH}$ and $\mathcal{UH}'$.

Round 1.

In this first round, each player U_i for $i = 1, \ldots, n$ starts by setting the partner identifier pid_i to $\{U_1, \ldots, U_n\}$. Then, each player U_i generates a pair (sk_i, vk_i) of secret and public keys for a signature scheme and a label $l_i = vk_i \,\|\, U_1 \,\|\, \ldots \,\|\, U_n$. Next, each player encrypts the joint group password pw using the encryption algorithm Enc with respect to the public key pk and label l_i using $r_i^{\mathtt{R}}$ as the randomness. Let $c_i^{\mathtt{R}}$ denote the resulting ciphertext (i.e., $c_i^{\mathtt{R}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathtt{R}})$). At the end of this round, each player U_i broadcasts the pair $(l_i, c_i^{\mathtt{R}})$.

Round 2.

In this second round, each player U_i for $i = 1, \ldots, n$ encrypts once more the joint group password pw using the encryption algorithm Enc with respect to the public key pk and label l_i using $r_i^{\mathtt{L}}$ as the randomness. Let $c_i^{\mathtt{L}}$ denote the resulting ciphertext (i.e., $c_i^{\mathtt{L}} = \mathsf{Enc}(pk, l_i, \mathsf{pw}; r_i^{\mathtt{L}})$). Next, each player U_i chooses a hash key $hk_i^{\mathtt{L}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function and then generates a projection key $phk_i^{\mathtt{L}}$ for it with respect to the pair $(c_{i-1}^{\mathtt{R}}, l_{i-1})$. That is, $phk_i^{\mathtt{L}} \xleftarrow{\$} \mathsf{ProjKG}(hk_i^{\mathtt{L}}, l_{i-1}, c_{i-1}^{\mathtt{R}})$. Here and in other parts of the protocol, the indices are taken modulo n. At the end of this round, each player U_i broadcasts the pair $(c_i^{\mathtt{L}}, phk_i^{\mathtt{L}})$.

Round 3.

In this round, player U_i first chooses two new hash keys hk_i and $hk_i^{\mathtt{R}}$ uniformly at random from $\mathbf{HK}(pk)$ for the smooth projective hash function. Next, player U_i generates two projection keys phk_i and $phk_i^{\mathtt{R}}$ for the hash keys hk_i and $hk_i^{\mathtt{R}}$, both with respect to the pair $(c_{i+1}^{\mathtt{L}}, l_{i+1})$. That is, $phk_i \xleftarrow{\$} \mathsf{ProjKG}(hk_i, l_{i+1}, c_{i+1}^{\mathtt{L}})$ and $phk_i^{\mathtt{R}} \xleftarrow{\$} \mathsf{ProjKG}(hk_i^{\mathtt{R}}, l_{i+1}, c_{i+1}^{\mathtt{L}})$. Then, player U_i computes a test master key $X_i^{\mathtt{R}} = K_{i+1}^{\mathtt{L}} \cdot K_i^{\mathtt{R}}$ for its successor, where $K_i^{\mathtt{L}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ and $K_i^{\mathtt{R}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{R}}, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$. Note that player U_i can compute $K_i^{\mathtt{R}}$ using $hk_i^{\mathtt{R}}$ and $K_{i+1}^{\mathtt{L}}$ using $phk_{i+1}^{\mathtt{L}}$ and the witness $r_i^{\mathtt{R}}$ to the fact that $c_i^{\mathtt{R}}$ is a valid encryption of pw with respect to pk and l_i. Finally, player U_i computes a test key $test_i^{\mathtt{R}} = \mathsf{UH}_1(X_i^{\mathtt{R}})$, sets $T_i^{\mathtt{R}} = U_i \,\|\, U_{i+1} \,\|\, c_i^{\mathtt{R}} \,\|\, c_{i+1}^{\mathtt{L}} \,\|\, phk_i \,\|\, phk_i^{\mathtt{R}} \,\|\, phk_{i+1}^{\mathtt{L}} \,\|\, test_i^{\mathtt{R}}$, and computes a signature $\sigma_i^{\mathtt{R}}$ on $T_i^{\mathtt{R}}$ using sk_i. At the end of this round, player U_i broadcasts the tuple $(phk_i, phk_i^{\mathtt{R}}, test_i^{\mathtt{R}}, \sigma_i^{\mathtt{R}})$.

Round 4.

In this round, each player U_i first verifies if the signature $\sigma_{i-1}^{\mathtt{R}}$ on the transcript $T_{i-1}^{\mathtt{R}}$ is correct using vk_{i-1}. If this check fails, then player U_i halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, player U_i computes the values $K_i^{\mathtt{L}}$ and $K_{i-1}^{\mathtt{R}}$, using the hash key $hk_i^{\mathtt{L}}$ and the projection key $phk_{i-1}^{\mathtt{R}}$ along with the witness $r_i^{\mathtt{L}}$ to the fact that $c_i^{\mathtt{L}}$ is a valid encryption of pw with respect to pk and l_i. That is, $K_i^{\mathtt{L}} = \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ and $K_{i-1}^{\mathtt{R}} = \mathsf{ProjHash}(phk_{i-1}^{\mathtt{R}}, c_i^{\mathtt{L}}, l_i, \mathsf{pw}, r_i^{\mathtt{L}})$. Next, player U_i computes the test master key $X_i^{\mathtt{L}} = K_i^{\mathtt{L}} \cdot K_{i-1}^{\mathtt{R}}$ for its predecessor and verifies if $test_{i-1}^{\mathtt{R}} =$

$\mathsf{UH}_1(X_i^{\mathtt{L}})$. Once again, if this test fails, then player U_i halts and sets $\mathsf{acc}_i = \mathsf{false}$. If this test succeeds, then player U_i computes a test key $test_i^{\mathtt{L}} = \mathsf{UH}_2(X_i^{\mathtt{L}})$ for its predecessor and an auxiliary key $X_i = K_i/K_{i-1}$, where $K_i \triangleq \mathsf{Hash}(hk_i, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$. More precisely, player U_i computes the value K_i using the hash key hk_i and the value K_{i-1} using the projection key phk_{i-1} along with the witness $r_i^{\mathtt{L}}$ to the fact that $c_i^{\mathtt{L}}$ is a valid encryption of pw with respect to pk and l_i. Finally, each player U_i broadcasts the pair $(X_i, test_i^{\mathtt{L}})$.

Round 5.

First, each player U_i checks whether $test_{i+1}^{\mathtt{L}} = \mathsf{UH}_2(X_i^{\mathtt{R}})$ and whether $\prod_{l=1}^{n} X_l = 1$. If any of these tests fails, then player U_i halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, each player U_i sets $T_j = vk_j \,\|\, U_j \,\|\, c_j \,\|\, phk_j \,\|\, phk_j^{\mathtt{L}} \,\|\, phk_j^{\mathtt{R}} \,\|\, X_j \,\|\, X_j^{\mathtt{L}}$ for $j = 1, \ldots, n$ and $T = T_1 \,\|\, \ldots \,\|\, T_n$ and then signs it using sk_i to obtain σ_i. Finally, each player U_i broadcasts σ_i.

Finalization.

Each player U_i checks for $j \neq i$ whether σ_j is a valid signature on T with respect to vk_j. If any of these checks fails, then player U_i halts and sets $\mathsf{acc}_i = \mathsf{false}$. Otherwise, player U_i sets $\mathsf{acc}_i = \mathsf{true}$ and computes the master key $MSK = \prod_{j=1}^{n} K_j = K_i^n \cdot X_{i+1}^{n-1} \cdot X_{i+2}^{n-2} \cdot \ldots \cdot X_{i+n-3}^{2} \cdot X_{i+n-1}$, and the session key $SK = \mathsf{UH}'(MSK)$. Each player U_i also sets the session identifier sid_i to T.

Observation.

Let $K_i \triangleq \mathsf{Hash}(hk_i, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$, $K_i^{\mathtt{R}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{R}}, c_{i+1}^{\mathtt{L}}, l_{i+1}, \mathsf{pw})$, and $K_i^{\mathtt{L}} \triangleq \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ denote temporary keys. In a normal execution of the protocol, the temporary keys K_i and $K_i^{\mathtt{R}}$ are known to both player U_i (who knows hk_i and $hk_i^{\mathtt{R}}$) and his successor U_{i+1} (who knows phk_i, $phk_i^{\mathtt{R}}$, and the witness $r_{i+1}^{\mathtt{L}}$ to the fact that $c_{i+1}^{\mathtt{L}}$ is a valid encryption of pw with respect to pk and l_{i+1}). Likewise, the temporary key $K_i^{\mathtt{L}}$ is known to both player U_i (who knows $hk_i^{\mathtt{L}}$) and his predecessor U_{i-1} (who knows $phk_i^{\mathtt{R}}$ and the witness $r_{i-1}^{\mathtt{R}}$ to the fact that $c_{i-1}^{\mathtt{R}}$ is a valid encryption of pw with respect to pk and l_{i-1}).

D.4.2 Correctness and Security

Correctness. In an honest execution of the protocol, it is easy to verify that all participants in the protocol will terminate by accepting and computing the same values for the partner identifier, session identifiers, and the session key. The session key in this case is equal to $\prod_{j=1}^{n} \mathsf{Hash}(hk_j, c_{j+1}, l_{j+1}, \mathsf{pw}) = \prod_{j=1}^{n} K_j$.

Security. As the following theorem shows, the $\mathcal{GPAKE}$ protocol described above and in Figure D.1 is a secure password-based authenticated group key exchange protocol as long as the primitives on which the protocol is based meet the appropriate security notion described in the theorem.

Theorem D.4.1 Let $\mathcal{LPKE}$ be a labeled encryption secure against chosen-ciphertext attacks, let $\mathcal{HASH}$ be a family of smooth projective hash functions, let $\mathcal{UH}$ and $\mathcal{UH}'$ be families of universal hash functions, and let $\mathcal{SIG}$ be a signature scheme that is unforgeable against chosen-message attacks. Let $\mathcal{GPAKE}$ denote the protocol built from these primitives as described above and let $\mathcal{A}$ be an adversary against $\mathcal{GPAKE}$. Then, the advantage function $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A}}(k)$ is only negligibly larger than $O(q/N)$, where q denotes the maximum number of different protocol instances to which $\mathcal{A}$ has asked *Send* queries and N is the dictionary size.

The proof of Theorem D.4.1 is in Appendix D.5. In it, we actually show that the security of our protocol is only negligibly larger than $(q_{send\text{-}1}+q_{send\text{-}2})/N$, where $q_{send\text{-}1}$ and $q_{send\text{-}2}$ represent the maximum number of *Send* queries that the adversary can ask with respect to the first and second round of communication and N is dictionary size. Even though we believe this security level is good enough for groups of small to medium sizes, it may not be sufficient in cases where the number of users in a group is large and the dictionary size is small. In the latter case, it would be desirable to have a scheme whose security is only negligibly larger than the number of sessions (and not protocol instances) over the size of the dictionary. Unfortunately, the latter cannot be achieved by our protocol as it is possible for an active adversary to test in the same session a number of passwords that is linear in the total number of users, for instance by playing the role of every other user.

D.4.3 Efficiency

Our protocol is quite efficient, only requiring a small amount of computation by each user. In what concerns encryption and hash computations, each user only has to perform 2 encryptions, 3 projection key generations, 3 hash computations, 3 projected hash computations, and 5 universal hash computations. The most expensive part of our protocol, which is linear in the group size, is the number of signature verifications and the master session key computation. While the latter computation can be improved by using algorithms for multi-exponentiations, the former can be improved by using two-time signature schemes.

It is worth mentioning here that, as done by Katz et al. [KMTG05] in the case of the KOY protocol [KOY01], one could also improve the efficiency of our protocol by using two different encryption schemes in the computation of the ciphertexts $c_i^{\mathtt{R}}$ and $c_i^{\mathtt{L}}$ broadcasted in the first and second rounds. While the computation of the ciphertexts $c_i^{\mathtt{R}}$ would require a CCA-secure labeled encryption scheme, the computation of the ciphertexts $c_i^{\mathtt{L}}$ would only require a CPA-secure encryption scheme. To show that this would actually be the case, one would need to modify the proofs of lemmas D.5.9 and D.5.13 in Appendix D.5. In the case of the proof of Lemma D.5.9 (which would rely on the CPA-secure encryption scheme), the decryption oracle can be avoided by using the secret key for the CCA-secure labeled encryption scheme to check the validity of a ciphertext $c_i^{\mathtt{R}}$. In the case of the proof of Lemma D.5.13 (which would rely on the CCA-secure labeled encryption scheme), one would only need the decryption oracle to check the validity of a ciphertext $c_i^{\mathtt{R}}$, since the validity of a ciphertext $c_i^{\mathtt{L}}$ would be checked using the secret key for the CPA-secure encryption scheme.

D.4.4 Future Work

One issue not addressed in the current paper is whether our protocol remains secure in the presence of *Corrupt* queries, through which the adversary can learn the values of the long-term secret keys held by a user. This is indeed a significant limitation of our security model which we expect to address in a future version of this paper. In fact, we do hope to be able to prove that our protocol achieves forward security according to the definition given in [KY03].

Acknowledgements

The authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT and by France Telecom R&D as part of the contract CIDRE, between France Telecom R&D and École normale supérieure.

D.5 Appendix: Proof of Theorem D.4.1

Notation.

Before proceeding with the proof of Theorem D.4.1, we define some basic notation and terminology that will be used in the proof. We start by classifying the *Send* queries into 6 categories, depending on the stage of the protocol to which the query is associated.

- $Send_0$. This query has the form $(U_i^{\langle j \rangle}, \{U_1, \ldots, U_n\})$ and denotes the initial *Send* query made to user instance $U_i^{\langle j \rangle}$ when executing the protocol in a group $\mathcal{G} = \{U_1, \ldots, U_n\}$ that contains user U_i. The output of this query is the pair $(l_i, c_i^{\mathtt{R}})$ that user instance $U_i^{\langle j \rangle}$ generates in the first round of communication.

- $Send_1$. This query has the form $(U_i^{\langle j \rangle}, \{(l_1, c_1^{\mathtt{R}}), \ldots, (l_n, c_n^{\mathtt{R}})\}, \{U_1, \ldots, U_n\})$ and denotes the second *Send* query to user instance $U_i^{\langle j \rangle}$. It returns the tuple $(phk_i^{\mathtt{L}}, c_i^{\mathtt{L}})$ that instance $U_i^{\langle j \rangle}$ would generate on input $\{(l_1, c_1^{\mathtt{R}}), \ldots, (l_n, c_n^{\mathtt{R}})\}$.

- $Send_2$. This query has the form $(U_i^{\langle j \rangle}, \{(phk_1^{\mathtt{L}}, c_1^{\mathtt{L}}), \ldots, (phk_n^{\mathtt{L}}, c_n^{\mathtt{L}})\}, \{U_1, \ldots, U_n\})$ and denotes the third *Send* query to user instance $U_i^{\langle j \rangle}$. It returns the tuple $(phk_i, phk_i^{\mathtt{R}}, test_i^{\mathtt{R}}, \sigma_i^{\mathtt{R}})$ that instance $U_i^{\langle j \rangle}$ would generate on input $\{(phk_1^{\mathtt{L}}, c_1^{\mathtt{L}}), \ldots, (phk_n^{\mathtt{L}}, c_n^{\mathtt{L}})\}$.

- $Send_3$. This query has the form $(U_i^{\langle j \rangle}, \{(phk_1, phk_1^{\mathtt{R}}, test_1^{\mathtt{R}}, \sigma_1^{\mathtt{R}}), \ldots, (phk_n, phk_n^{\mathtt{R}}, test_n^{\mathtt{R}}, \sigma_n^{\mathtt{R}})\}, \{U_1, \ldots, U_n\})$ denotes the fourth *Send* query to user instance $U_i^{\langle j \rangle}$. It returns the pair $(X_i, test_i^{\mathtt{L}})$ that instance $U_i^{\langle j \rangle}$ would generate on input $\{(phk_1, phk_1^{\mathtt{R}}, test_1^{\mathtt{R}}, \sigma_1^{\mathtt{R}}), \ldots, (phk_n, phk_n^{\mathtt{R}}, test_n^{\mathtt{R}}, \sigma_n^{\mathtt{R}})\}$.

- $Send_4$. This query has the form $(U_i^{\langle j \rangle}, \{(X_1, test_1^{\mathtt{L}}), \ldots, (X_n, test_n^{\mathtt{L}})\}, \{U_1, \ldots, U_n\})$ denotes the fifth *Send* query to user instance $U_i^{\langle j \rangle}$. It returns the signature σ_i that instance $U_i^{\langle j \rangle}$ would generate on input $\{(X_1, test_1^{\mathtt{L}}), \ldots, (X_n, test_n^{\mathtt{L}})\}$.

- $Send_5(U_i^{\langle j \rangle}, \{\sigma_1, \ldots, \sigma_n\}, \{U_1, \ldots, U_n\})$. This denotes the last *Send* query to user instance $U_i^{\langle j \rangle}$. This query forces instance $U_i^{\langle j \rangle}$ to either halt and reject (setting $\mathsf{acc}_i^j = \mathsf{false}$) or to halt, accept (setting $\mathsf{acc}_i^j = \mathsf{true}$), and compute a session key. It has no output.

We say a message is t-`oracle-generated` if that message has been generated by an oracle in round t of some session. We say that a ciphertext-label pair $(c^{\mathtt{R}}, l = vk \,\|\, U_1 \,\|\, \ldots \,\|\, U_n)$ is `valid for` $U_i^{\langle j \rangle}$ if $c^{\mathtt{R}}$ is a valid encryption of the password $\mathsf{pw}_{\mathcal{G}}$ with respect to the public key pk and the label l, and $\mathsf{pid}_i^j = \mathcal{G} = \{U_1, \ldots, U_n\}$.

Proof of Theorem D.4.1.

Let $\mathcal{A}$ be an adversary against our password-based group key exchange protocol $\mathcal{GPAKE}$. The goal of proof is to show that the advantage of $\mathcal{A}$ in breaking the semantic security of $\mathcal{GPAKE}$ is only negligibly better than $q_{send\text{-}1} + q_{send\text{-}2}$ over the size of the dictionary. To do so, our proof uses a sequence of hybrid experiments, the first of which corresponds to the actual attack. For each hybrid experiment $\mathbf{Hyb}_n$, we define an event $\textsc{Succ}_n$ corresponding to the case in which the adversary $\mathcal{A}$ correctly guesses the bit b involved in the *Test* query. Likewise, we also define the corresponding advantage of the adversary $\mathcal{A}$ in each of these experiments to be $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_n}(k) = 2 \cdot \Pr[\,\textsc{Succ}_n\,] - 1$.

Hybrid experiment $\mathbf{Hyb}_0$. This first experiment corresponds to a real attack, in which all the parameters, such as the public parameters in the common reference string and the passwords

associated with each group of users, are chosen as in the actual scheme. By definition, the advantage of an adversary $\mathcal{A}$ in guessing the bit b involved in the *Test* query in this experiment is exactly the same as in the real attack.

$$\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_0}(k) \triangleq \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A}}(k) \tag{D.1}$$

Hybrid experiment $\mathbf{Hyb}_1$. In this experiment, we change the simulation of the *Execute* oracle so that the values K_i, $K_i^{\mathtt{L}}$, and $K_i^{\mathtt{R}}$ for $i = 1, \ldots, n$ are chosen uniformly at random from $\mathbf{G}$, where n is the size of the group $\mathcal{G}$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. This is due to pseudorandomness property of the smooth projective hash function.

Lemma D.5.1 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_1}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_0}(k)\right| \leq \text{neg}(k)$.

Proof: To prove this lemma, we show how to construct an adversary $\mathcal{D}$ against the pseudorandomness property of the smooth projective hash function from an adversary $\mathcal{A}$ capable of distinguishing the current experiment from the previous one.

To do so, we recall that the adversary $\mathcal{D}$ is given a public key pk for a labeled encryption scheme and access to two oracles: $\textsc{Enc}(\cdot,\cdot)$ and $\textsc{Hash}(\cdot,\cdot,\cdot)$. The oracle $\textsc{Enc}(\cdot,\cdot)$ receives as input a message m and a label l and outputs a ciphertext c, which is an encryption of message m with respect to the public key pk and the label l. The oracle $\textsc{Hash}(\cdot,\cdot,\cdot)$ receives as input a label l, a message m, and a ciphertext c. If the ciphertext c is not the result of a previous query (l, m) to the oracle Enc, then it outputs $\perp$. Otherwise, the oracle Hash computes a hash key hk via $hk \xleftarrow{\$} \mathsf{HashKG}(pk)$ and a projection hash key phk as $phk \xleftarrow{\$} \mathsf{ProjKG}(hk, l, c)$. Then, Hash computes the output g of the hash function in one of two ways depending on the experiment in which it is. In experiment $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$, Hash sets g to $\mathsf{Hash}(hk, l, m, c)$. In experiment $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$, it sets g to a random value in $\mathbf{G}$. Finally, Hash outputs (phk, g). The goal of adversary $\mathcal{D}$ is to tell the exact experiment with which he is dealing. By the pseudorandomness property of a smooth projective hash function, no adversary $\mathcal{D}$ should be able to distinguish the experiment $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$ from the experiment $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$ with non-negligible probability.

We construct an adversary $\mathcal{D}$ as follows. First, $\mathcal{D}$ uses pk to initialize the common reference string exactly as in the experiment $\mathbf{Hyb}_0$. Then, whenever $\mathcal{A}$ asks a *Send* or *Execute* query with respect to a group $\mathcal{G}$ for which no password has been defined, then $\mathcal{D}$ chooses a password $\mathsf{pw}_{\mathcal{G}}$ for that group uniformly at random from the dictionary **Dict**. Then, $\mathcal{D}$ continues to simulate all of $\mathcal{A}$'s oracles exactly as in experiment $\mathbf{Hyb}_0$ except when computing the values $c_i^{\mathtt{R}}$, $c_i^{\mathtt{L}}$, phk_i, $phk_i^{\mathtt{L}}$, $phk_i^{\mathtt{R}}$, K_i, $K_i^{\mathtt{L}}$, and $K_i^{\mathtt{R}}$ in a *Execute* query for a group $\mathcal{G}$. To compute the latter values, $\mathcal{D}$ makes uses of its Enc and Hash oracles. That is, $\mathcal{D}$ first computes $c_i^{\mathtt{R}} = \textsc{Enc}(l_i, \mathsf{pw}_{\mathcal{G}})$ and $c_i^{\mathtt{L}} = \textsc{Enc}(l_i, \mathsf{pw}_{\mathcal{G}})$ for $i = 1, \ldots, |\mathcal{G}|$. Next, $\mathcal{D}$ queries its oracle Hash three times for each i in $\{1, \ldots, |\mathcal{G}|\}$ on inputs $(l_{i-1}, c_{i-1}^{\mathtt{R}})$, $(l_{i+1}, c_{i+1}^{\mathtt{L}})$, and $(l_{i+1}, c_{i+1}^{\mathtt{L}})$, to obtain respectively the values $(phk_i^{\mathtt{L}}, K_i^{\mathtt{L}})$, $(phk_i^{\mathtt{R}}, K_i^{\mathtt{R}})$, and (phk_i, K_i). Finally, $\mathcal{D}$ continues the simulation of the *Execute* query using the values above exactly as in the experiment $\mathbf{Hyb}_0$. At the end of the simulation, $\mathcal{D}$ outputs the same guess as $\mathcal{A}$.

One can easily see that, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$, then its simulation of the *Execute* oracle is performed exactly as in experiment $\mathbf{Hyb}_0$. Moreover, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$, then its simulation of the *Execute* oracle is performed exactly as in experiment $\mathbf{Hyb}_1$. Thus, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$ is exactly $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_1}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_0}(k)$.

Thus, the proof of Lemma D.5.1 follows from the pseudorandomness property of the family of smooth projective hash functions. ▌

Hybrid experiment $\mathbf{Hyb}_2$. In this experiment, we change again the simulation of the *Execute* oracle so that the master key MSK_i computed by each player in an *Execute* query is chosen uniformly at random from the group $\mathbf{G}$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The arguments used to prove this lemma are in fact information theoretic.

Lemma D.5.2 $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_2}(k) = \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_1}(k)$.

Proof: The proof of Lemma D.5.2 uses an argument similar to the one used by Katz and Yung in their proof of Burmester-Desmedt protocol [KY03]. First, we note that in the simulation of *Execute* oracle in experiment $\mathbf{Hyb}_1$, the values K_i for $i = 1, \ldots, |\mathcal{G}|$ are all chosen at random in $\mathbf{G}$. Second, let g denote a generator for $\mathbf{G}$. We note that, from the transcript T that the adversary receives as output for an *Execute* query, the values K_i are constrained by the following $|\mathcal{G}|$ equations.

$$\begin{aligned} \log_g X_1 &= \log_g K_1 - \log_g K_{|\mathcal{G}|} \\ &\vdots \\ \log_g X_{|\mathcal{G}|} &= \log_g K_{|\mathcal{G}|} - \log_g K_{|\mathcal{G}|-1}. \end{aligned}$$

Of these equations, only $|\mathcal{G}| - 1$ are linearly independent. Finally, the master key computed by the players defines an additional equation

$$\log_g MSK = \sum_{i=1}^{|\mathcal{G}|} \log_g K_i.$$

Since the last equation is linearly independent of the previous ones, the master secret key MSK_i that each player in the group $\mathcal{G}$ computes in an *Execute* query is independent of the transcript T that the adversary $\mathcal{A}$ sees. Thus, no computationally unbounded adversary $\mathcal{A}$ can tell experiment $\mathbf{Hyb}_2$ apart from $\mathbf{Hyb}_1$. As a result, for any $\mathcal{A}$, $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_2}(k) = \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_1}(k)$. ▌

Hybrid experiment $\mathbf{Hyb}_3$. In this experiment, we change once more the simulation of the *Execute* oracle so that the session key SK_i computed by each player in an *Execute* query is chosen uniformly at random in $\{0,1\}^l$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.3 follows easily from the properties of the family of universal hash functions $\mathcal{UH}'$, which guarantees that its output is statistically close to uniform in $\{0,1\}^l$ when given a random value in $\mathbf{G}$ as input.

Lemma D.5.3 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_3}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_2}(k)\right| \leq \text{neg}(k)$.

Hybrid experiment $\mathbf{Hyb}_4$. In this experiment, we change one last time the simulation of the *Execute* oracle so that the password $\mathsf{pw}_{\mathcal{G}}$ associated with a group $\mathcal{G}$ is no longer used. More specifically, when simulating an *Execute* query, the ciphertexts c_i^{R} and c_i^{L} that each player $U_i \in \mathcal{G}$ computes becomes the encryption of a dummy password $\mathsf{pw}'_{\mathcal{G}} \neq \mathsf{pw}_{\mathcal{G}}$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.4 follows from the semantic security of the labeled encryption scheme $\mathcal{LPKE}$.

Lemma D.5.4 $|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_4}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_3}(k)| \leq \text{neg}(k)$.

Proof: To prove this lemma, we show how to construct an adversary $\mathcal{D}$ against the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$ from an adversary $\mathcal{A}$ capable of distinguishing the current experiment from the previous one. We recall that the adversary $\mathcal{D}$ is given a public key pk for the labeled encryption scheme and access to two oracles: the Left-or-Right encryption oracle $\textsc{Enc}(\cdot,\cdot,\cdot)$ and a decryption oracle $\textsc{Dec}(\cdot,\cdot)$. The latter, however, is not needed in this part of the proof.

We construct an adversary $\mathcal{D}$ as follows. First, $\mathcal{D}$ uses pk to initialize the common reference string exactly as in the experiment $\mathbf{Hyb}_3$. Then, whenever $\mathcal{A}$ asks a *Send* or *Execute* query with respect to a group $\mathcal{G}$ for which no password has been defined, then $\mathcal{D}$ first chooses a password $\mathsf{pw}_{\mathcal{G}}$ for that group uniformly at random from the dictionary **Dict**. At this time, $\mathcal{D}$ also chooses a dummy password $\mathsf{pw}'_{\mathcal{G}} \neq \mathsf{pw}_{\mathcal{G}}$ of the appropriate length. Next, $\mathcal{D}$ continues to simulate all of $\mathcal{A}$'s oracles exactly as in experiment $\mathbf{Hyb}_3$ except when computing the ciphertext values $c_i^{\mathtt{R}}$ and $c_i^{\mathtt{L}}$ in a *Execute* query for a user group $\mathcal{G}$. To compute the latter values, $\mathcal{D}$ makes uses of its Left-or-Right encryption oracle $\textsc{Enc}$. More precisely, $\mathcal{D}$ computes $c_i^{\mathtt{R}} = \textsc{Enc}(l_i, \mathsf{pw}_{\mathcal{G}}, \mathsf{pw}'_{\mathcal{G}})$ and $c_i^{\mathtt{L}} = \textsc{Enc}(l_i, \mathsf{pw}_{\mathcal{G}}, \mathsf{pw}'_{\mathcal{G}})$ for $i = 1, \ldots, |\mathcal{G}|$. Finally, $\mathcal{D}$ continues the simulation of the remaining part of the *Execute* query exactly as in the experiment $\mathbf{Hyb}_3$, choosing the session key SK_i of each player U_i and the intermediate values $K_i^{\mathtt{L}}$, $K_i^{\mathtt{R}}$, and K_i uniformly at random in their respective groups. At the end of the simulation, $\mathcal{D}$ outputs the same guess as $\mathcal{A}$.

One can easily see that, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$, then its simulation of the *Execute* oracle is performed exactly as in experiment $\mathbf{Hyb}_3$, since in this case the Left-or-Right encryption oracle $\textsc{Enc}$ always returns the encryption of the actual password. Moreover, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$, then its simulation of the *Execute* oracle is performed exactly as in experiment $\mathbf{Hyb}_4$, since in this case the Left-or-Right encryption oracle $\textsc{Enc}$ always returns the encryption of a dummy password. Thus, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$ is exactly $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_4}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_3}(k)$. Thus, the proof of Lemma D.5.4 follows from the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$. ▌

Hybrid experiment $\mathbf{Hyb}_5$. In this new experiment, we change the simulation of the *Send* oracle of an instance $U_i^{\langle j \rangle}$ whenever the latter receives an 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ (the one that should have been created by its predecessor) in $Send_1$ query. Let $U_{i'}^{\langle j' \rangle}$ be the instance that created this ciphertext (note that i' may or may not be equal to $i-1$). More precisely, if the instance $U_i^{\langle j \rangle}$ does not reject after a $Send_3$ query (by setting $\mathsf{acc}_i^j = \mathsf{false}$), then the values K_{i-1} and $K_{i-1}^{\mathtt{R}}$ are chosen uniformly at random from $\mathbf{G}$. Remember from the protocol that instance $U_i^{\langle j \rangle}$ halts and sets $\mathsf{acc}_i^j = \mathsf{false}$ whenever the signature $\sigma_{i-1}^{\mathtt{R}}$ on the transcript $T_{i-1}^{\mathtt{R}}$ with respect to vk_{i-1} is not valid or the test session key $test_{i-1}^{\mathtt{R}}$ is not correct. Furthermore, if $i' = i-1$ and the values $c_{i-1}^{\mathtt{R}}$ and $c_i^{\mathtt{L}}$ seen by instances $U_i^{\langle j \rangle}$ and $U_{i'}^{\langle j' \rangle}$ are the same, then the values $K_{i'}$ and $K_{i'}^{\mathtt{L}}$ for instance $U_{i'}^{\langle j' \rangle}$ are also chosen uniformly at random. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.5 follows from the security properties of the smooth projective hash function family $\mathcal{HASH}$.

Lemma D.5.5 $|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_5}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_4}(k)| \leq \text{neg}(k)$.

Proof: The proof of Lemma D.5.5 has two cases depending on whether or not the 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ is `valid for` $U_i^{\langle j \rangle}$. We observe here that ciphertext $c_{i-1}^{\mathtt{R}}$ is `valid for` $U_i^{\langle j \rangle}$ if it

has been created by an instance $U_{i'}^{\langle j' \rangle}$, forwarded to instance $U_i^{\langle j \rangle}$, and it holds that $\mathsf{pid}_{i'}^{j'} = \mathsf{pid}_i^j$ and $i' = i - 1$.

$c_{i-1}^{\mathtt{R}}$ IS `invalid for` $U_i^{\langle j \rangle}$. The proof in this case follows easily from the smoothness property of the family of smooth projective hash functions, which says that if $c_{i-1}^{\mathtt{R}}$ is *not* a valid encryption of the password $\mathsf{pw}_{\mathcal{G}}$ with respect to the public-key pk and label l_{i-1}, then the hash value $K_i^{\mathtt{L}} = \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ is *statistically* close to uniform in $\mathbf{G}$. Likewise, the master key $X_i^{\mathtt{L}} = K_i^{\mathtt{L}} \cdot K_{i-1}^{\mathtt{R}}$ used to check the test session key of its predecessor is also *statistically* close to uniform in $\mathbf{G}$. As a result, except with negligible probability, the instance $U_i^{\langle j \rangle}$ will halt and reject (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving the test value $test_{i-1}^{\mathtt{R}}$. Hence, the difference in the advantage between the current experiment and previous one due to this case is a negligible function of the security parameter.

$c_{i-1}^{\mathtt{R}}$ IS `valid for` $U_i^{\langle j \rangle}$. The proof of this case follows from pseudorandomness property of the family of smooth projective hash functions. To prove so, we show how to construct an adversary $\mathcal{D}$ against the pseudorandomness property of the family $\mathcal{HASH}$ from an adversary $\mathcal{A}$ capable of distinguishing the current experiment from the previous one when $c_{i-1}^{\mathtt{R}}$ is `valid for` $U_i^{\langle j \rangle}$.

We construct an adversary $\mathcal{D}$ as follows. First, $\mathcal{D}$ uses pk to initialize the common reference string exactly as in the experiment $\mathbf{Hyb}_4$. Then, whenever $\mathcal{A}$ asks a *Send* or *Execute* query with respect to a group $\mathcal{G}$ for which no password has been defined, then $\mathcal{D}$ chooses a password $\mathsf{pw}_{\mathcal{G}}$ for that group uniformly at random from the dictionary **Dict**. Then, $\mathcal{D}$ continues to simulate all of $\mathcal{A}$'s oracles exactly as in experiment $\mathbf{Hyb}_4$ except when an instance $U_i^{\langle j \rangle}$ receives a valid 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ from an instance $U_{i-1}^{\langle j' \rangle}$. In the latter, instead of computing the ciphertext $c_i^{\mathtt{L}}$ as in the original protocol, $\mathcal{D}$ obtains $c_i^{\mathtt{L}}$ by making a call to its ENC oracle using the label l_i and password $\mathsf{pw}_{\mathcal{G}}$ as the input. Then, if the adversary $\mathcal{A}$ correctly forwards the ciphertext $c_i^{\mathtt{L}}$ to instance $U_{i-1}^{\langle j' \rangle}$ in a $Send_2$ query, then $\mathcal{D}$ makes two calls to his HASH oracle using $c_i^{\mathtt{L}}$ and label l_i as input to obtain respectively the values $(phk_{i-1}^{\mathtt{L}}, K_{i-1}^{\mathtt{L}})$, and (phk_{i-1}, K_{i-1}) instead of computing these values by itself. $\mathcal{D}$ also uses the value $K_{i-1}^{\mathtt{L}}$ and K_{i-1} to compute $test_{i-1}^{\mathtt{R}}$ in round 3 and the value $X_{i-1}^{\mathtt{L}}$ in round 4. Next, if instance $U_i^{\langle j \rangle}$ receives the projection keys $phk_{i-1}^{\mathtt{L}}$ and phk_{i-1} in a $Send_2$ query that $\mathcal{D}$ obtained from the HASH oracle, then $\mathcal{D}$ also uses the values $K_{i-1}^{\mathtt{L}}$ and K_{i-1} to compute $X_i^{\mathtt{L}}$ and X_i. If instance $U_i^{\langle j \rangle}$ does not receive the projection keys $phk_i^{\mathtt{L}}$ and phk_i in a $Send_2$ query, then it halts and sets $\mathsf{acc}_i^j = \mathsf{false}$. Apart from these changes, no other modification is made to the simulation. At the end of the simulation, $\mathcal{D}$ outputs the same guess as $\mathcal{A}$.

To analyze the success probability of $\mathcal{D}$, first notice that if instance $U_i^{\langle j \rangle}$ receives projection keys $phk_{i-1}^{\mathtt{L}}$ and phk_{i-1} different from those outputted by the HASH oracle, then $U_i^{\langle j \rangle}$ rejects with overwhelming probability due to the security of the signature scheme $\mathcal{SIG}$. This is because the case we are considering is the one in which the ciphertext $c_{i-1}^{\mathtt{R}}$ is 1-`oracle-generated` and `valid for` $U_i^{\langle j \rangle}$. Thus, the only way for $\mathcal{A}$ to send projections keys $phk_{i-1}^{\mathtt{L}}$ and phk_{i-1} different from those computed by instance $U_{i-1}^{\langle j' \rangle}$ is for the latter to forge the signature $\sigma_{i-1}^{\mathtt{R}}$. A formal reduction to the security of the signature scheme $\mathcal{SIG}$ in this case is straight-forward and omitted here.

Next, notice that, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$, then its simulation of the *Send* oracle is performed exactly as in experiment $\mathbf{Hyb}_4$. This is because in this case, whenever an instance $U_i^{\langle j \rangle}$ receives a valid 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$, the values for K_{i-1} and $K_{i-1}^{\mathtt{R}}$ that $U_i^{\langle j \rangle}$ computes come from the HASH oracle and are thus correct. Moreover, if both $U_i^{\langle j \rangle}$

and $U_{i-1}^{\langle j' \rangle}$ see the same transcript $T_{i-1}^{\mathtt{R}}$, then $U_{i-1}^{\langle j' \rangle}$ also computes the same values for K_{i-1} and $K_{i-1}^{\mathtt{R}}$ (except when $\mathcal{A}$ succeeds in forging a signature). Finally, notice that, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$, then its simulation of the *Send* oracle is performed almost exactly as in experiment $\mathbf{Hyb}_5$ since the values for K_{i-1} and $K_{i-1}^{\mathtt{R}}$ that $U_i^{\langle j \rangle}$ computes when receiving a valid 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ are random in this case. The only difference occurs when $\mathcal{A}$ succeeds in forging the signature $\sigma_{i-1}^{\mathtt{R}}$ of instance $U_{i-1}^{\langle j' \rangle}$. Thus, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{h-prf-real}}_{\mathcal{HASH},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{h-prf-random}}_{\mathcal{HASH},\mathcal{D}}(k)$ is negligibly close to $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_5}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_4}(k)$. Hence, the difference in the advantage between the current experiment and previous one due to this case is a negligible function of the security parameter.

Since in both cases, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter, the proof of Lemma D.5.5 follows. ▌

Hybrid experiment $\mathbf{Hyb}_6$. In this new experiment, we change once again the simulation of the *Send* oracle of an instance $U_i^{\langle j \rangle}$ whenever the latter receives an 1-`oracle-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ in a $Send_1$ query so that $U_i^{\langle j \rangle}$ computes the ciphertext $c_i^{\mathtt{L}}$ using a dummy password $\mathsf{pw}'_{\mathcal{G}}$ that is different from the password $\mathsf{pw}_{\mathcal{G}}$ associated with the group $\mathcal{G}$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.6 follows from the chosen-plaintext security of the labeled encryption scheme $\mathcal{LPKE}$.

Lemma D.5.6 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_6}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_5}(k)\right| \leq \text{neg}(k)$.

Proof: The proof of this lemma is similar to that of Lemma D.5.4 and follows easily from the chosen-plaintext security of the labeled encryption scheme $\mathcal{LPKE}$. To construct the adversary $\mathcal{D}$ for $\mathcal{LPKE}$ from an adversary $\mathcal{A}$ capable of distinguishing the current experiment from the previous one, we proceed as follows. First, $\mathcal{D}$ uses pk to initialize the common reference string exactly as in the experiment $\mathbf{Hyb}_5$. Then, whenever $\mathcal{A}$ asks a $Send_0$ or *Execute* query with respect to a group $\mathcal{G}$ for which no password has been defined, then $\mathcal{D}$ first chooses a password $\mathsf{pw}_{\mathcal{G}}$ for that group uniformly at random from the dictionary **Dict** as well as a dummy password $\mathsf{pw}'_{\mathcal{G}} \neq \mathsf{pw}_{\mathcal{G}}$ of the appropriate length. also chooses a dummy password $\mathsf{pw}'_{\mathcal{G}} \neq \mathsf{pw}_{\mathcal{G}}$ of the appropriate length. Next, $\mathcal{D}$ continues to simulate all of $\mathcal{A}$'s oracles exactly as in experiment $\mathbf{Hyb}_5$ except when $\mathcal{A}$ makes a $Send_1$ query to an $U_i^{\langle j \rangle}$ using a 1-`oracle-generated` value for the ciphertext $c_{i-1}^{\mathtt{R}}$. To simulate the latter query, $\mathcal{D}$ computes $c_i^{\mathtt{L}}$ as $\textsc{Enc}(l_i, \mathsf{pw}_{\mathcal{G}}, \mathsf{pw}'_{\mathcal{G}})$ with the help of its Left-or-Right encryption oracle Enc. One should note here that this change is only possible because the values K_{i-1} and $K_{i-1}^{\mathtt{R}}$ that $U_i^{\langle j \rangle}$ may need to compute in the sessions are chosen at random from $\mathbf{G}$ since the previous experiment. No other change is made to the simulation. At the end of the simulation, $\mathcal{D}$ outputs the same guess as $\mathcal{A}$.

One can easily see that, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$, then its simulation of the $Send_1$ oracle is performed exactly as in experiment $\mathbf{Hyb}_5$, since Enc always returns the encryption of the actual password in this case. Moreover, whenever $\mathcal{D}$ is in experiment $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$, then its simulation of the $Send_1$ oracle is performed exactly as in experiment $\mathbf{Hyb}_6$, since Enc always returns the encryption of a dummy password in this case. Thus, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$ is exactly $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_6}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_5}(k)$. Thus, the Lemma D.5.6 follows from the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$. ▌

Hybrid experiment $\mathbf{Hyb}_7$. In this experiment, we change the simulation so that, whenever an instance $U_i^{\langle j \rangle}$ receives a *valid* `adversarially-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ in a $Send_1$ query, we halt the simulation and consider the adversary $\mathcal{A}$ successful. No other change is made to simulation. Clearly, if such ciphertext $c_{i-1}^{\mathtt{R}}$ is never sent by $\mathcal{A}$, then the current and the previous experiments are identical. On the other hand, if the adversary $\mathcal{A}$ does happen to send such a valid ciphertext, then $\mathcal{A}$ is considered successful. Therefore, as the following lemma states, the advantage of $\mathcal{A}$ in the current experiment is greater or equal to that in the previous experiment.

Lemma D.5.7 $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_6}(k) \leq \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_7}(k)$.

Hybrid experiment $\mathbf{Hyb}_8$. In this experiment, we change the simulation so that, whenever an instance $U_i^{\langle j \rangle}$ receives a *invalid* `adversarially-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ in a $Send_1$ query, then $U_i^{\langle j \rangle}$ always chooses the value $K_i^{\mathtt{L}}$ uniformly at random from $\mathbf{G}$. Moreover, it always halts and rejects (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving a $Send_3$ query from $\mathcal{A}$. Everything else remains the same. As the following lemma shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter.

Lemma D.5.8 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_8}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_7}(k)\right| \leq \text{neg}(k)$.

Proof: The proof of Lemma D.5.8 follows easily from the smoothness property of smooth projective hash function family $\mathcal{HASH}$, which says that if $c_{i-1}^{\mathtt{R}}$ is *not* a valid encryption of the password $\mathsf{pw}_{\mathcal{G}}$ with respect to the public-key pk and label l_{i-1}, then the hash value $K_i^{\mathtt{L}} = \mathsf{Hash}(hk_i^{\mathtt{L}}, c_{i-1}^{\mathtt{R}}, l_{i-1}, \mathsf{pw})$ is *statistically* close to uniform in $\mathbf{G}$. Likewise, the master key $X_i^{\mathtt{L}} = K_i^{\mathtt{L}} \cdot K_{i-1}^{\mathtt{R}}$ used to check the test session key of its predecessor is also *statistically* close to uniform in $\mathbf{G}$. As a result, except with negligible probability, the instance $U_i^{\langle j \rangle}$ will halt and reject (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving the test value $test_{i-1}^{\mathtt{R}}$. Hence, the difference in the advantage between the current experiment and previous one due to this case is a negligible function of the security parameter. ∎

Hybrid experiment $\mathbf{Hyb}_9$. In this experiment, we change once again the simulation of the *Send* oracle of an instance $U_i^{\langle j \rangle}$ whenever the latter receives a *invalid* `adversarially-generated` ciphertext $c_{i-1}^{\mathtt{R}}$ in a $Send_1$ query so that the latter computes the ciphertext $c_i^{\mathtt{L}}$ using dummy password $\mathsf{pw}'_{\mathcal{G}}$ that is different from the $\mathsf{pw}_{\mathcal{G}}$ associated the group $\mathcal{G}$. Everything else remains the same. As the following lemma shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.9 follows from the semantic security of the labeled encryption scheme $\mathcal{LPKE}$.

Lemma D.5.9 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_9}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_8}(k)\right| \leq \text{neg}(k)$.

Proof: The proof of this lemma is similar to that of Lemma D.5.6 and follows easily from the chosen-ciphertext security of the labeled encryption scheme $\mathcal{LPKE}$. As in that case, we are only able to make this change to the simulation because instance $U_i^{\langle j \rangle}$ always chooses the value $K_i^{\mathtt{L}}$ uniformly at random from $\mathbf{G}$ and because it always halts and rejects (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving a $Send_3$ query from $\mathcal{D}$. The only difference between the proof of Lemma D.5.6 and the present one is that here we need to make use of the decryption oracle for $\mathcal{LPKE}$ to find out if $c_{i-1}^{\mathtt{R}}$ is a valid encryption of the group password $\mathsf{pw}_{\mathcal{G}}$ with respect to the public key pk and the label l_{i-1}. This is because the ciphertext $c_{i-1}^{\mathtt{R}}$ is generated by the adversary in this case. Like in the proof of Lemma D.5.6, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$ and

$\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$ is exactly $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_9}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_8}(k)$. Thus, Lemma D.5.9 follows from the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$. ▮

Hybrid experiment $\mathbf{Hyb}_{10}$. In this new experiment, we change the simulation of the $Send_2$ oracle of an instance $U_i^{\langle j \rangle}$ so that, whenever the latter receives an 2-`oracle-generated` ciphertext c_{i+1}^{L}, the values K_i and K_i^{R} are chosen uniformly at random from $\mathbf{G}$. As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.10 is omitted here since it follows easily from the smoothness property of smooth projective hash function family $\mathcal{HASH}$ as, since the previous any 2-`oracle-generated` ciphertext c_{i+1}^{L} is always an encryption of a dummy password in this case and hence not valid.

Lemma D.5.10 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{10}}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_9}(k)\right| \leq \text{neg}(k)$.

Hybrid experiment $\mathbf{Hyb}_{11}$. In this experiment, we change the simulation of the $Send_2$ oracle of an instance $U_i^{\langle j \rangle}$ so that, whenever an instance $U_i^{\langle j \rangle}$ receives a *valid* `adversarially-generated` ciphertext c_{i+1}^{L} in a $Send_2$ query, we halt the simulation and consider the adversary $\mathcal{A}$ successful. No other change is made to simulation. Clearly, if such ciphertext c_{i+1}^{L} is never sent by $\mathcal{A}$, then the current and the previous experiments are identical. On the other hand, if the adversary $\mathcal{A}$ does happen to send such a valid ciphertext, then $\mathcal{A}$ is considered successful. Therefore, as the following lemma states, the advantage of $\mathcal{A}$ in the current experiment is greater or equal to that in the previous experiment.

Lemma D.5.11 $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{10}}(k) \leq \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{11}}(k)$.

Hybrid experiment $\mathbf{Hyb}_{12}$. In this experiment, we change once again the simulation of the $Send_2$ oracle of an instance $U_i^{\langle j \rangle}$ so that, whenever an instance $U_i^{\langle j \rangle}$ receives a *invalid* `adversarially-generated` ciphertext c_{i+1}^{L} in a $Send_2$ query, then $U_i^{\langle j \rangle}$ always chooses the values K_i^{R} and K_i uniformly at random from $\mathbf{G}$. Moreover, it always halts and rejects (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving a $Send_4$ query from $\mathcal{A}$. Everything else remains the same. As the following lemma shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter.

Lemma D.5.12 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{12}}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{11}}(k)\right| \leq \text{neg}(k)$.

Proof: As in the proof of Lemma D.5.8, Lemma D.5.12 follows easily from the smoothness property of smooth projective hash function family $\mathcal{HASH}$, which says that if c_{i+1}^{L} is *not* a valid encryption of the password $\mathsf{pw}_{\mathcal{G}}$ with respect to the public-key pk and label l_{i+1}, then the hash values $K_i^{\mathsf{R}} = \mathsf{Hash}(hk_i^{\mathsf{R}}, c_{i+1}^{\mathsf{L}}, l_{i+1}, \mathsf{pw}_{\mathcal{G}})$ and $K_i = \mathsf{Hash}(hk_i, c_{i+1}^{\mathsf{L}}, l_{i+1}, \mathsf{pw}_{\mathcal{G}})$ are *statistically* close to uniform in $\mathbf{G}$. Likewise, the master key $X_i^{\mathsf{R}} = K_i^{\mathsf{R}} \cdot K_{i+1}^{\mathsf{L}}$ used to check the test session key of its successor is also *statistically* close to uniform in $\mathbf{G}$. As a result, except with negligible probability, the instance $U_i^{\langle j \rangle}$ will halt and reject (by setting $\mathsf{acc}_i^j = \mathsf{false}$) after receiving the test value $test_{i+1}^{\mathsf{L}}$. Hence, the difference in the advantage between the current experiment and previous one due to this case is a negligible function of the security parameter. ▮

Hybrid experiment $\mathbf{Hyb}_{13}$. In this experiment, we change the simulation of the $Send_0$ oracle of an instance $U_i^{\langle j \rangle}$ so that the latter computes the ciphertext c_i^{R} using dummy password $\mathsf{pw}'_{\mathcal{G}}$ that is different from the $\mathsf{pw}_{\mathcal{G}}$ associated the group $\mathcal{G}$. Everything else remains the same. As the following lemma shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. The proof of Lemma D.5.13 follows from the semantic security of the labeled encryption scheme $\mathcal{LPKE}$.

Lemma D.5.13 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{13}}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{12}}(k)\right| \leq \text{neg}(k)$.

Proof: The proof of this lemma is similar to that of Lemma D.5.9 and follows easily from the chosen-ciphertext security of the labeled encryption scheme $\mathcal{LPKE}$. As in that case, we are only able to make this change because at this point instance $U_i^{\langle j \rangle}$ no longer needs to know the randomness $r_i^{\mathtt{R}}$ used to create the ciphertext $c_i^{\mathtt{R}}$ to be able to compute the test master key $X_i^{\mathtt{R}}$ since the latter is always a random value in $\mathbf{G}$ (this is because $K_i^{\mathtt{R}}$ is always chosen uniformly at random from $\mathbf{G}$). As in the proof of Lemma D.5.9, we also need access to a decryption oracle for $\mathcal{LPKE}$ to be to verify whether `adversarially-generated` ciphertexts $c_{i-1}^{\mathtt{R}}$ and $c_{i+1}^{\mathtt{L}}$ are valid encryptions of the group password $\mathsf{pw}_{\mathcal{G}}$, the first with respect to the pair (pk, l_{i-1}) and the second with respect to the pair (pk, l_{i+1}). Like in the proof of Lemma D.5.9, the probability that $\mathcal{D}$ distinguishes between experiments $\mathbf{Exp}^{\text{lpke-ind-cca-1}}_{\mathcal{LPKE},\mathcal{D}}(k)$ and $\mathbf{Exp}^{\text{lpke-ind-cca-0}}_{\mathcal{LPKE},\mathcal{D}}(k)$ is exactly $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{13}}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{12}}(k)$. Thus, Lemma D.5.13 follows from the chosen-ciphertext indistinguishability of $\mathcal{LPKE}$. ∎

Hybrid experiment $\mathbf{Hyb}_{14}$. In this experiment, we change the simulation of the $Send_5$ oracle so that the master key MSK computed by an instance $U_i^{\langle j \rangle}$ is chosen uniformly at random from the group $\mathbf{G}$ whenever $U_i^{\langle j \rangle}$ accepts (by setting $\mathsf{acc}_i^j = \mathsf{true}$). As the lemma below shows, the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter.

Lemma D.5.14 $\left|\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{14}}(k) - \mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{13}}(k)\right| \leq \text{neg}(k)$.

Proof: The proof of Lemma D.5.14 follows from the unforgeability of the signature scheme $\mathcal{SIG}$ and uses arguments similar to those in the proof of Lemma D.5.2. To prove this, first, we note that an instance $U_i^{\langle j \rangle}$ that accepts only does so after verifying that the instances of users U_{i-1} and U_{i+1} in $\mathcal{G}$ that are next to it actually know the correct password (this is done in rounds 3 and 4) and after verifying the correctness of all the signatures that it receives in the last round of communication. Thus, whenever an instance $U_i^{\langle j \rangle}$ does not halt and reject, it must be the case that its neighbors are in fact oracle instances not played by the adversary (since we always halt the simulation when the adversary produces a valid ciphertext). Moreover, the keys K_i and K_{i-1} computed by instance $U_i^{\langle j \rangle}$ and by its predecessor are chosen at random from $\mathbf{G}$. Second, we note that in all sessions in which the adversary plays an active role, with very overwhelming probability, he causes all the oracles instances in that session to halt and reject. While the instances that are next to $\mathcal{A}$ reject in round 3 or 4 because $\mathcal{A}$ has sent an invalid ciphertext in one of the first two rounds, the other ones reject with overwhelming probability because $\mathcal{A}$ does not succeed in forging the signatures of those instances that have rejected before the last round of communication (this requires a formal reduction to the security of the signature scheme $\mathcal{SIG}$, which is straight-forward and omitted here). This holds because the signature of a user always guarantees the validity of the verification keys associated with his successor and predecessor. Thus, attacks in which the adversary sends different verification keys to different users will always cause at least one of signatures in the ensemble to be invalid.

Putting everything together, we can conclude that, whenever an instance $U_i^{\langle j \rangle}$ accepts, its session partners are in fact oracle instances not played by the adversary and that all keys K_t for $t = 1, \ldots, |\mathcal{G}|$ are random values in $\mathbf{G}$. As a result, the values X_t for $t = 1, \ldots, |\mathcal{G}|$ outputted by the $Send_4$ oracles define exactly $|\mathcal{G}|$ equations, of which $|\mathcal{G}| - 1$ are linearly independent, as in the proof of Lemma D.5.2. Since in round 5, each instance broadcasts its own signature of the transcript, it must be the case that, if $U_i^{\langle j \rangle}$ accepts after a $Send_5$ query, with high probability

it has received the correct values X_t computed by each partner instance of U_t in that session. If that is not the case, then it is straight-forward to build an adversary capable of breaking the security of the signature scheme $\mathcal{SIG}$. Finally, since the master key computed by instance $U_i^{\langle j \rangle}$ defines an additional equation, which is linearly independent of the other $|\mathcal{G}| - 1$ equations that were defined by the values $X_1, \ldots, X_{|\mathcal{G}|}$, its value is independent of the transcript T that the adversary $\mathcal{A}$ sees. It follows that the difference in the advantage between the current experiment and previous one is a negligible function of the security parameter. ▮

To conclude the proof, we first notice that, since the session keys of all accepting instances are chosen at random and since all session partners that accept end up computing the same session key (due to the security of the signature scheme $\mathcal{SIG}$), the advantage of $\mathcal{A}$ in the current experiment is 0 when $\mathcal{A}$ does not generate a valid ciphertext round 1 or 2. Second, in the current experiment, all oracle instances are simulated using dummy passwords, $\mathcal{A}$'s view of the protocol is independent of the passwords that are chosen for each group of users. Finally, since each ciphertext uniquely defines a password, we have that the probability that any given adversarially generated ciphertext is valid is at most $1/N$, where N is the size of the dictionary. As the number of adversarially generated ciphertexts is bounded by $q_{send\text{-}1} + q_{send\text{-}2}$, the advantage of $\mathcal{A}$ in the current experiment is only negligibly larger than $(q_{send\text{-}1} + q_{send\text{-}2})/N$.

Lemma D.5.15 $\mathbf{Adv}^{\text{ake-ind}}_{\mathcal{GPAKE},\mathcal{A},\mathbf{Hyb}_{14}}(k) \leq (q_{send\text{-}1} + q_{send\text{-}2})/N + \text{neg}(k)$.

By combining all the lemmas above, one easily obtains the announced result in Theorem D.4.1.

Identity-Based Cryptography

Appendix E:

Wildcarded Identity-Based Encryption, Journal of Cryptology, 24(1):42–82
M. Abdalla, J. Birkett, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, J. Schuldt, and N. Smart
This article introduces the notion of identity-based encryption with wildcards, which is a generalization of identity-based encryption that allows a sender to encrypt messages to a whole range of receivers whose identities match a certain pattern. In addition to proposing the new primitive, this paper also presents several efficient implementations which are shown to be secure under an appropriate security notion.

Appendix F:

Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions, Journal of Cryptology, 21(3):350–391
M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi
This article revisits the notion of public-key encryption with keyword search, also known as searchable encryption, and provides new security definitions for these schemes which take false positives into account. It also provides generic transformations to and from anonymous identity-based encryption schemes and suggests a few extensions, such as public-key encryption with temporary keyword search, and identity-based encryption with keyword search.

Appendix G:

Robust Encryption, TCC 2010
Michel Abdalla, Mihir Bellare, and Gregory Neven
This article formalizes the notion of robust encryption which reflects the difficulty of producing a ciphertext that is valid under two different encryption keys. In addition to the formalization in the public-key and identity-based settings, this paper also assesses the robustness of specific encryption schemes in the literature, providing simple patches for some that lack the property. Moreover, it also shows that the robustness of anonymous identity-based encryption schemes is closely related to issue of false positives of searchable encryption schemes.

Appendix E

Wildcarded Identity-Based Encryption

Journal of Cryptology, 24(1):42–82
[ABC+11] with J. Birkett, D. Catalano, A. Dent, J. Malone-Lee, G. Neven, J. Schuldt, and N. Smart

Abstract : *In this paper we introduce a new primitive called identity-based encryption with wildcards, or WIBE for short. It allows a sender to encrypt messages to a whole range of receivers whose identities match a certain pattern. This pattern is defined through a sequence of fixed strings and wildcards, where any string can take the place of a wildcard in a matching identity. Our primitive can be applied to provide an intuitive way to send encrypted email to groups of users in a corporate hierarchy. We propose a full security notion and give efficient implementations meeting this notion under different pairing-related assumptions, both in the random oracle model and in the standard model.*

E.1 Introduction

The concept of identity-based cryptography was introduced by Shamir as early as in 1984 [Sha85], and the same paper proposed an identity-based signature scheme. However, it took nearly twenty years for an efficient identity-based encryption (IBE) scheme to be proposed. In 2000 and 2001 respectively Sakai, Ohgishi and Kasahara [SOK00] and Boneh and Franklin [BF03] proposed IBE schemes based on elliptic curve pairings. Also, in 2001 Cocks proposed a system based on the quadratic residuosity problem [Coc01].

One of the main application areas proposed for IBE is that of email encryption. In this scenario, given an email address, one can encrypt a message to the owner of the email address without needing to obtain an authentic copy of the owner's public key first. In order to decrypt the email the recipient must authenticate itself to a trusted authority who generates a private key corresponding to the email address used to encrypt the message.

E.1.1 Identity-Based Encryption with Wildcards

Our work is motivated by the fact that many email addresses correspond to groups of users rather than single individuals. Consider the scenario where there is some kind of organisational

hierarchy. Take as an example an organisation called ECRYPT which is divided into virtual labs, say AZTEC and STVL. In addition, these virtual labs are further subdivided into working groups WG1, WG2 and WG3. Finally, each working group may consist of many individual members. There are several extensions of the IBE primitive to such a hierarchical setting (HIBE) [HL02, GS02]. The idea is that each level can issue keys to users on the level below. For example the owner of the ECRYPT key can issue decryption keys for ECRYPT.AZTEC and ECRYPT.STVL.

Suppose that we wish to send an email to all the members of the AZTEC.WG1 working group, which includes the personal addresses

- ECRYPT.AZTEC.WG1.Nigel,
- ECRYPT.AZTEC.WG1.Dario,
- ECRYPT.AZTEC.WG1.John.

Given a standard HIBE one would have to encrypt the message to each user individually. To address this limitation we introduce the concept of *identity-based encryption with wildcards* (WIBE). The way in which decryption keys are issued is exactly as in a standard HIBE scheme; what differs is encryption. Our primitive allows the encrypter to replace any component of the recipient identity with a *wildcard* so that any identity matching the *pattern* can decrypt. Denoting wildcards by `*`, in the example above the encrypter would use the identity

- ECRYPT.AZTEC.WG1.$*$

to encrypt to all members of the AZTEC.WG1 group.

It is often suggested that identity strings should be appended with the date so as to add timeliness to the message, and so try to mitigate the problems associated with key revocation. Using our technique we can now encrypt to a group of users, with a particular date, by encrypting to an identity of the form

- ECRYPT.AZTEC.WG1.$*$.22Oct2006

for example. Thus any individual in the group

- ECRYPT.AZTEC.WG1

with a decryption key for 22nd October 2006 will be able to decrypt.

As another example, take a hierarchy of email addresses at academic institutions of the form

- `name@department.university.edu`,

i.e., the email address of John Smith working at the computer science department of Some State University would be `johnsmith@cs.ssu.edu`. Using our primitive, one can send encrypted email to everyone in the computer science department at Some State University by encrypting to identity `*@cs.ssu.edu`, to everyone at SSU by encrypting to `*@*.ssu.edu`, to all computer scientists at any institution by encrypting to `*@cs.*.edu`, or to all system administrators in the university by encrypting to `sysadmin@*.ssu.edu`.

E.1.2 Our Contributions

In this paper, we introduce the primitive of identity-based encryption with wildcards, or a WIBE for short. We define appropriate security notions under chosen-plaintext and chosen-ciphertext attack, and present the first instantiations of this primitive. In more detail, we present the

syntax and security notions in Section E.3. To illustrate the relationship between WIBEs and other identity-based primitives, we show how WIBE schemes can be built from HIBE schemes and from fuzzy identity-based encryption schemes.

As is the case for most public-key and identity-based encryption schemes, the non-hybrid WIBE schemes can only be used to encrypt relatively short messages, typically about 160 bits. To encrypt longer messages, one will have to resort to hybrid techniques: the sender uses the WIBE to encrypt a fresh symmetric key K and encrypts the actual message under the key K. The basic construction has been used within the cryptographic community for years, dating back to the work of Blum and Goldwasser in 1984 [BG85], but its security for the case of public-key encryption was not properly analysed until the work of Cramer and Shoup [CS03]. One would intuitively expect these results to extend to the case of WIBEs, which is indeed the case. We present the syntax for a WIB-KEM in Section E.4, along with the composition theorem which proves that the combination of a secure WIB-KEM and a secure DEM results in a secure WIBE scheme.

We also give several constructions for a WIBE scheme, classified according to their security guarantees. We first present the Boneh-Boyen WIBE (BB-WIBE – see Section E.5.1) and the Boneh-Boyen-Goh WIBE (BBG-WIBE – see Section E.5.2). These schemes are IND-CPA secure in the selective identity model and do not require random oracles to be proven secure, although we do require random oracles in order to prove their security in the full (non-selective-identity) model (see Section E.5.4). We also present the Waters WIBE scheme (see Section E.5.3) which is secure in the non-selective-identity IND-CPA setting without random oracles.

The range of IND-CPA WIBE schemes available makes selection difficult. The Waters WIBE scheme has the best security guarantees, but the worst performance. In particular, the number of elements in the master public key depends upon the maximum length of an identity, which is typically of the order of 160 bits. Hence, even with a small number of levels, the size of the master public key can be prohibitive. Both the BB-WIBE scheme and the BBG-WIBE scheme have better performance characteristics, but their security (in the non-selective-identity model) depends on random oracles. Furthermore, the BBG-WIBE scheme reduces to the less-studied L-BDHI assumption, but has the best performance characteristics.

The construction of IND-CCA secure WIBE schemes is more difficult. We present two generic transformations from an IND-CPA scheme into an IND-CCA scheme. The first transformation is based on the Canetti-Halevi-Katz transform (see Section E.6.1) which builds an L-level IND-WID-CCA secure WIBE from an $(L+1)$-level IND-WID-CPA WIBE. The disadvantage of our construction compared to the original CHK transform is that our construction always encrypts messages under patterns of length $L+1$. This often increases the space and time complexity of the scheme in practical situations (as the worst performance characteristic are often obtained for "full-length" patterns). The approach we present in this paper is different from the approach given in the ePrint version of [ACD+06], which requires using $2L+2$ levels as opposed to $L+1$. We thank the anonymous referee for helping guide us to this improvement.

Our second transform is based on Dent's construction of a KEM (see Section E.6.2). This converts a weakly secure (one-way) WIBE scheme into an IND-CCA secure WIB-KEM, but requires the random oracle model in order to prove its security. We note that one-way security is implied by IND-CPA security (for sufficiently large messages spaces). Consequently, we can use any of the IND-CPA constructions given in Section E.5 to build an IND-CCA secure scheme.

In [BDNS07] we also presented a WIB-KEM in the standard model based on the Kiltz-Galindo HIB-KEM from [KG09]. Due to our improved CPA to CCA transform described above this is no longer as efficient as the transformed Waters WIBE, hence we do not consider the Kiltz-Galindo WIB-KEM in this paper.

An overview of all the schemes we present is given in Table E.1 and Table E.2.

Scheme	$\|mpk\|$	$\|d\|$	$\|C\|$	Decrypt	Assumption	RO
Generic	$\|mpk_{HIBE}\|$	$2^L \cdot \|d_{HIBE}\|$	$\|C_{HIBE}\|$	$\mathsf{Decrypt}_{HIBE}$	IND-HID-CPA HIBE	No
BB-WIBE	$2L+3$	$L+1$	$2L+2$	$L+1$	BDDH	Yes
BBG-WIBE	$L+4$	$L+2$	$L+3$	2	L-BDHI	Yes
Waters-WIBE	$(n+1)L+3$	$L+1$	$(n+1)L+2$	$L+1$	BDDH	No

Table E.1: Efficiency comparison between our CPA-secure schemes. We compare the generic scheme of Section E.3.3, the Waters-WIBE scheme of Section E.5.3, the BB-WIBE scheme of Section E.5.1, the BBG-WIBE scheme of Section E.5.2, and the Waters-WIBE scheme of Section E.5.3. The schemes are compared in terms of master number of elements in the public key ($|mpk|$), number of elements in the user secret key ($|d|$), number of element in the ciphertext ($|C|$), number of pairing operations required for decryption (Decrypt), the security assumption under which the scheme is proved secure, and whether this proof is in the random oracle model or not. The generic construction does not introduce any random oracles, but if the security proof of the HIBE scheme is in the random oracle model, then the WIBE obviously inherits this property. L is the maximal hierarchy depth and n is the bit length of an identity string. Figures are worst-case values, usually occurring for identities at level L with all-wildcard ciphertexts.

Scheme	$\|mpk\|$	$\|d\|$	$\|C\|$	Encap	Decap	Security loss
OML(BB-WIBE)	$2L+5$	$L+1$	$2L+3$	$2L+4$	$L+2$	q_H^{L+1}
OW(BB-WIBE)	$2L+3$	$L+1$	$2L+2$	$2L+2$	$L+1$	q_H^{L}
OML(BBG-WIBE)	$L+4$	$L+1$	$L+3$	$L+3$	2	q_H^{L+1}
OW(BBG-WIBE)	$L+4$	$L+1$	$L+3$	$L+3$	2	q_H^{L}
OML(Waters)	$(n+1)(L+1)+3$	$L+1$	$(n+1)L+3$	$(n+1)L+3$	$L+2$	$(2nq_K)^{L+1}$

Table E.2: Efficiency comparison between our CCA-secure schemes. The BB-WIBE scheme is the IND-WID-CPA scheme given in E.5.1; the BBG-WIBE scheme is the IND-WID-CPA scheme given in E.5.2; the Waters scheme is the IND-WID-CPA scheme given in E.5.3. The $OML(\cdot)$ transformation refers to the (one more level) generic CCA-secure construction of a CCA-secure WIBE from a CPA-secure WIBE presented in Section E.6.1. The $OW(\cdot)$ transformation is our random-oracle based construction of a WIB-KEM scheme from a CPA-secure WIBE presented in Section E.6.2. We compare the schemes in terms of number of elements in the master public key ($|mpk|$), number of elements in the user secret key ($|d|$), number of elements in the ciphertext ($|C|$), number of exponentiations required for key encapsulation (Encap), number of pairings required for key decapsulation (Decap), and the dominant factor lost in the security reduction to the underlying assumption. L is the maximal hierarchy depth and n is the bit length of an identity string. The values q_H and q_K refer to the number of queries made by an adversary to the random oracle and key derivation oracle, respectively.

E.2 A Recap on Various Primitives

In this section, we recall basic notation and known results on different primitives that we will be using throughout this paper. In particular, we will recall several constructions of Hierarchical Identity-Based Encryption schemes (HIBEs) upon which out Wildcarded Identity-Based Encryption schemes (WIBEs) are based.

E.2.1 Basic Notation

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers. Let ε be the empty string. If $n \in \mathbb{N}$, then $\{0,1\}^n$ denotes the set of n-bit strings and $\{0,1\}^*$ is the set of all finite bit strings. If

$s = (s_1, \ldots, s_n)$ is an ordered sequence of n elements of some set and $0 \leq \ell \leq n$, then $s_{\leq \ell}$ is the ordered sequence consisting of the first ℓ elements of s, i.e. $s_{\leq \ell} = (s_1, \ldots, s_\ell)$. Furthermore, if ID is an n-bit string, then we set

$$[ID_i] = \{1 \leq j \leq n : \text{ the } j^{th} \text{ bit of } ID_i \text{ is one}\}.$$

If S is a finite set, then $y \xleftarrow{\$} S$ denotes the assignment to y of a randomly chosen element of the set S. If $\mathcal{A}$ is a deterministic algorithm, then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output $\mathcal{A}$ when run on the input x. If $\mathcal{A}$ is a randomised algorithm, then $y \xleftarrow{\$} \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ on the input x when the algorithm is run with fresh random coins.

E.2.2 Hash Functions

A hash function is a family of maps $F_k : \mathcal{IP} \to \mathcal{OP}$ index by a keyspace $\mathcal{K}$ in which the output space $\mathcal{OP}$ is finite. The input space $\mathcal{IP}$ may be finite or infinite. Additionally, the key space may be empty or non-empty. There are many security properties that can be ascribed to a hash function. We will only need to consider one security property at this time (although we will introduce further security notions in later sections and may model these hash functions as random oracles).

Definition E.2.1 A (t, ϵ)-adversary $\mathcal{A}$ against the second pre-image resistance property of a family of hash functions $F_k : \mathcal{IP} \to \mathcal{OP}$ with a finite input space $\mathcal{IP}$ is an algorithm that runs in time at most t and has advantage at least ϵ, where the adversary's advantage is defined to be:

$$\Pr[x \neq y \wedge F_k(x) = F_k(y) : x \xleftarrow{\$} \mathcal{IP}; k \xleftarrow{\$} \mathcal{K}; y \xleftarrow{\$} \mathcal{A}(k, x)].$$

E.2.3 One-Time Signature Schemes

In order to amplify the security of a HIBE/WIBE (from IND-CPA security to IND-CCA security) we will make use of a one-time signature scheme. A one-time signature scheme is a triple of algorithms $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$. The key generation algorithm SigGen outputs signing and verification keys (sk, vk) for the signature scheme. The signing algorithm takes as input a signing key sk and a message $m \in \{0,1\}^*$, and outputs a signature $\sigma \in \{0,1\}^*$. The verification algorithm takes as input a verification key vk, a message $m \in \{0,1\}^*$ and a signature $\sigma \in \{0,1\}^*$, and outputs either $\top$ (indicating a valid signature) or $\bot$ (indicating an invalid signature). For correctness, we require that for all key pairs (sk, vk), messages $m \in \{0,1\}^*$, and signatures $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$, we have that $\mathsf{Verify}(vk, m, \sigma) = \top$ with probability one.

The security notion for a one-time unforgeable signature scheme is captured by the following game played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger:

1. The challenger generates a key pair $(sk^*, vk^*) \xleftarrow{\$} \mathsf{SigGen}$.
2. The adversary runs $\mathcal{A}_1$ on input vk^*. The adversary outputs a message m^* and some state information *state*.
3. The challenger computes $\sigma^* \xleftarrow{\$} \mathsf{Sign}(sk^*, m^*)$.
4. The adversary runs $\mathcal{A}_2$ on σ^* and *state*. The adversary outputs a message signature pair (m, σ).

The adversary wins the game if $\mathsf{Verify}(vk^*, m, \sigma) = \top$ and $(m, \sigma) \neq (m^*, \sigma^*)$. The adversary's advantage is defined to be $\Pr[\mathcal{A} \text{ wins}]$.

Definition E.2.2 A (t, ϵ)-adversary against the one-time unforgeability of the signature scheme is an algorithm that runs in time t and has advantage at least ϵ in winning the above game.

E.2.4 Bilinear Maps and Related Assumptions

Let $\mathbb{G}, \mathbb{G}_T$ be multiplicative groups of prime order p with an admissible map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. By admissible we mean that the map is bilinear, non-degenerate and efficiently computable. Bilinearity means that for all $a, b \in \mathbb{Z}_p$ and all $g \in \mathbb{G}$ we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$. By non-degenerate we mean that $\hat{e}(g, g) = 1$ if and only if $g = 1$.

In such a setting we can define a number of computational problems. The first we shall be interested in is called the bilinear decisional Diffie-Hellman (BDDH) problem [Jou04]: given a tuple (g, g^a, g^b, g^c, T), the problem is to decide whether $T = \hat{e}(g, g)^{abc}$ or whether it is a random element of $\mathbb{G}_T$. More formally, we define the following game between an adversary $\mathcal{A}$ and a challenger. The challenger first chooses a random generator $g \xleftarrow{\$} \mathbb{G}^*$, random integers $a, b, c \xleftarrow{\$} \mathbb{Z}_p$, a random element $T \xleftarrow{\$} \mathbb{G}_T$, and a random bit $\beta \xleftarrow{\$} \{0, 1\}$. If $\beta = 1$ it feeds $\mathcal{A}$ the tuple $(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc})$ as input; if $\beta = 0$ it feeds $\mathcal{A}$ the tuple (g, g^a, g^b, g^c, T) as input. The adversary $\mathcal{A}$ must then output its guess β' for β. The adversary has advantage ϵ in solving the BDDH problem if

$$\Big| \Pr\Big[\mathcal{A}(g, g^a, g^b, g^c, \hat{e}(g, g)^{abc}) = 1\Big] - \Pr\Big[\mathcal{A}(g, g^a, g^b, g^c, T) = 1\Big] \Big| \geq \epsilon,$$

where the probabilities are over the random choice of g, a, b, c, T and over the random coins of $\mathcal{A}$.

Definition E.2.3 A (t, ϵ)-adversary $\mathcal{A}$ against the BDDH problem is an algorithm that runs in time at most t and has advantage at least ϵ.

We note that throughout this paper we will assume that the time t of an adversary includes its code size, in order to exclude trivial "lookup" adversaries.

A second problem we will use in our constructions is the ℓ-bilinear Diffie-Hellman Inversion (ℓ-BDHI) problem [BB04a, MSK02]. The problem is to compute $\hat{e}(g, g)^{1/\alpha}$ for random $g \xleftarrow{\$} \mathbb{G}^*$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p$ given $g, g^\alpha, \ldots, g^{(\alpha^\ell)}$. The decisional variant of this problem is to distinguish $\hat{e}(g, g)^{1/\alpha}$ from a random element of $\mathbb{G}_T$. We say that adversary $\mathcal{A}$ has advantage ϵ in solving the decisional ℓ-BDHI problem if

$$\Big| \Pr\Big[\mathcal{A}(g, g^\alpha, \ldots, g^{(\alpha^\ell)}, \hat{e}(g, g)^{1/\alpha}) = 1\Big] - \Pr\Big[\mathcal{A}(g, g^\alpha, \ldots, g^{(\alpha^\ell)}, T) = 1\Big] \Big| \geq \epsilon \ ,$$

where the probability is over the random choice of $g \xleftarrow{\$} \mathbb{G}^*$, $\alpha \xleftarrow{\$} \mathbb{Z}_p$, $T \xleftarrow{\$} \mathbb{G}_T$, and the coins of $\mathcal{A}$.

Definition E.2.4 A (t, ϵ)-adversary against the decisional ℓ-BDHI problem is an algorithm that runs in time at most t and has advantage at least ϵ in the above game.

We note that the BDDH problem is a weaker assumption than the ℓ-BDHI assumption. Hence, all other things being equal schemes which are based on the BDDH assumption are to be preferred to ones based on the ℓ-BDHI assumption. However, our most efficient constructions are based on the ℓ-BDHI assumption as opposed to the BDDH assumption. As the two assumptions are very different in nature, it is hard to compare precisely various schemes; indeed the comparison would depends on the readers view with respect to the intepretation of exact security results and the view of the relative hardness of the two underlying problems.

E.2.5 Hierarchical Identity-Based Encryption

An identity-based encryption (IBE) scheme is a tuple of algorithms ($\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt}$) providing the following functionality. The trusted authority runs Setup to generate a master key pair (mpk, msk). It publishes the master public key mpk and keeps the master secret key msk private. When a user with identity ID wishes to become part of the system, the trusted authority generates a decryption key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$, and sends this key over a secure and authenticated channel to the user. To send an encrypted message m to the user with identity ID, the sender computes the ciphertext $C \xleftarrow{\$} \mathsf{Encrypt}(mpk, ID, m)$, which can be decrypted by the user as $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$. We refer to [BF03] for details on the security definitions for IBE schemes.

In this paper, we are more interested in the concept of Hierarchical Identity-Based Encryption (HIBE) [HL02, GS02]. In a HIBE scheme, users are organised in a tree of depth L, with the root being the master trusted authority. The identity of a user at level $0 \leq \ell \leq L$ in the tree is given by a vector $ID = (ID_1, \ldots, ID_\ell) \in (\{0,1\}^*)^\ell$. A HIBE scheme is a tuple of algorithms ($\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt}$) providing the same functionality as in an IBE scheme, except that a user $ID = (ID_1, \ldots, ID_\ell)$ at level ℓ can use its own secret key d_{ID} to generate a secret key for any of its children $ID' = (ID_1, \ldots, ID_\ell, ID_{\ell+1})$ via $d_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(d_{ID}, ID_{\ell+1})$. Note that by iteratively applying the KeyDer algorithm, user ID can derive secret keys for any of its descendants $ID' = (ID_1, \ldots, ID_{\ell+\delta})$, $\delta \geq 0$. We will occasionally use the overloaded notation

$$d_{ID'} \xleftarrow{\$} \mathsf{KeyDer}(d_{ID}, (ID_{\ell+1}, \ldots, ID_{\ell+\delta}))$$

to denote this process. The secret key of the root identity at level 0 is $d_\epsilon \leftarrow msk$. Encryption and decryption are the same as for IBE, but with vectors of bit strings as identities instead of ordinary bit strings.

The security of a HIBE scheme is defined through the following IND-HID-CPA game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary outputs two equal-length messages (m_0, m_1) and a challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$, along with some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, ID^*, m_\beta)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary outputs a bit β'.

In most cases, we will suppress the *state* information passed between adversary algorithms and simply assume that all necessary details are passed from one algorithm to the next. The adversary wins the game if $\beta = \beta'$ and it never queries the key derivation oracle with any ancestor identity of ID^*, i.e. any identity $ID = (ID_1^*, \ldots, ID_\ell^*)$ where $\ell \leq \ell^*$. The adversary's advantage is defined to be equal to $|2 \cdot \Pr[\,\mathcal{A} \text{ wins}\,] - 1|$.

Definition E.2.5 A (t, q_K, ϵ)-adversary against the IND-HID-CPA security of a HIBE scheme is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, and has advantage at least ϵ in winning the IND-HID-CPA game described above.

The IND-HID-CCA security game is identical to the IND-HID-CPA security game with the exception that in the IND-HID-CCA security game the adversary additionally has access to a decryption oracle that, on input of a ciphertext C and an identity ID, returns the decryption $m \leftarrow \mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle with any ancestor identity of ID^*, and it never queries the decryption oracle with the pair (C^*, ID^*) after the challenge ciphertext is computed.

Definition E.2.6 A (t, q_K, q_D, ϵ)-adversary against the IND-HID-CCA security of the HIBE scheme is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-HID-CCA game described above.

In a *selective-identity* (sID) attack [BB04a], the adversary has to output the challenge identity ID^* at the very beginning of the game, before even seeing the master public key. In other words, the adversary is considered to be a triple $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_0$ simply outputs the challenge identity (and some state information to be passed to $\mathcal{A}_1$). The definitions for IND-HID-CPA and IND-HID-CCA security are otherwise identical to those above. In the random oracle model [BR93], all algorithms, as well as the adversary, have access to a random oracle mapping arbitrary bit strings onto a range that possibly depends on the master public key. All above security definitions then take an extra parameter q_H denoting the adversary's maximum number of queries to the random oracle.

We now recap on the main efficient HIBE constructions in the literature, namely the HIBE schemes of Waters (W-HIBE), Boneh-Boyen (BB-HIBE), and Boneh-Boyen-Goh (BBG-HIBE).

E.2.6 The Boneh-Boyen HIBE

In this section, we present a variant of the HIBE scheme by Boneh and Boyen [BB04a]. In this scheme, we assume that identities are vectors of elements of $\mathbb{Z}_p$ – if necessary this can be achieved by applying a collision-resistant hash function $h : \{0,1\}^* \to \mathbb{Z}_p$ to binary identities before applying the scheme. The scheme is described in Figure E.1. The main difference between the original HIBE scheme of [BB04a] and our variant above is that our scheme uses a different value $u_{i,1}$ for each level, while the original scheme uses the same value u_1 for all levels. Adding wildcard functionality to the original scheme would require us to include u_1^r in the ciphertext, but this ruins security as it can be used to change the identity for which a ciphertext is encrypted.

For completeness, we prove the security of this new HIBE scheme, despite its similarities to scheme of Boneh and Boyen [BB04a].

Theorem E.2.7 If there exists a (t, q_K, ϵ)-adversary against the IND-sHID-CPA security of the BB-HIBE (with hierarchy depth L) then there exists a (t', ϵ')-adversary against the BDDH problem in $\mathbb{G}$, where $\epsilon' \geq \epsilon - q_K/p$ and $t' \leq t + O(L \cdot q_K \cdot t_{exp})$ and t_{exp} is the maximum time for an exponentiation in $\mathbb{G}$ and p is the order of $\mathbb{G}$.

Proof: The present proof follows very closely the proof of security for the original scheme in [BB04a]. As before, we assume that there exist an adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND-sID-CPA-security of the BB-HIBE scheme and then we show how to efficiently build another adversary $\mathcal{B}$ that, using $\mathcal{A}$ as a subroutine, manages to solve the BDDH problem in $\mathbb{G}$.

Algorithm $\mathcal{B}$ first receives as input a random tuple $(g, A = g^a, B = g^b, C = g^c, Z)$ and its goal is to determine whether $Z = \hat{e}(g,g)^{abc}$ or $\hat{e}(g,g)^z$ for a random element z in $\mathbb{Z}_p$. Algorithm $\mathcal{B}$ should output 1 if $Z = \hat{e}(g,g)^{abc}$ and 0 otherwise. Algorithm $\mathcal{B}$ works as follows.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1 \ldots L, j = 0, 1$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,1})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm $\mathsf{KeyDer}(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot \left(u_{\ell+1,0} \cdot u_{\ell+1,1}^{ID_{\ell+1}}\right)^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm $\mathsf{Encrypt}(mpk, ID, m)$:
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1, \ldots, \ell$ do
$C_{2,i} \leftarrow \left(u_{i,0} \cdot u_{i,1}^{ID_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm $\mathsf{Decrypt}(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.1: The Boneh-Boyen HIBE scheme.

Initialisation. Algorithm $\mathcal{B}$ starts by running algorithm $\mathcal{A}_0$, which responds with the challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$ where $0 \leq \ell^* \leq L$. If $\ell^* = L$ then $\mathcal{B}$ sets $\tilde{ID}^* \leftarrow ID^*$. Otherwise, $\mathcal{B}$ randomly generates $ID_{\ell^*+1}^*, \ldots, ID_L^* \xleftarrow{\$} \mathbb{Z}_p$ and sets $\tilde{ID}^* \leftarrow (ID_1^*, \ldots, ID_L^*)$.

Setup. To generate the systems parameters, $\mathcal{B}$ first sets $g_1 \leftarrow g$, $h_1 \leftarrow A$, and $g_2 \leftarrow B$. Algorithm $\mathcal{B}$ then chooses $\alpha_{1,0}, \ldots, \alpha_{L,0}, \alpha_{1,1}, \ldots, \alpha_{L,1} \xleftarrow{\$} \mathbb{Z}_p^*$ at random and sets $u_{i,0} \leftarrow g_1^{\alpha_{i,0}} \cdot h_1^{-ID_i^* \alpha_{i,1}}$ and $u_{i,1} \leftarrow h_1^{\alpha_{i,1}}$ for $i = 1, \ldots, L$. $\mathcal{B}$ defines the master public key to be $mpk \leftarrow (g_1, h_1, g_2, u_{1,0}, \ldots, u_{L,0}, u_{1,1}, \ldots, u_{L,1})$. Note that the corresponding master secret key $msk = g_2^a$ is unknown to $\mathcal{B}$.

Phase 1. $\mathcal{B}$ runs $\mathcal{A}_1$ on input mpk. If $\mathcal{A}_1$ makes a key derivation oracle query on $ID = (ID_1, \ldots, ID_\ell)$, where $ID_i \in \mathbb{Z}_p$ and $\ell \leq L$ then ID cannot be a prefix of ID^*. Hence, if ID is a prefix of $\tilde{ID}^*$ then $\mathcal{A}$ aborts; we let E be the event that this occurs. Otherwise, let j be the smallest index such that $ID_j \neq \tilde{ID}_j^*$. To reply to this query, $\mathcal{B}$ first computes the key for identity $ID' = (ID_1, \ldots, ID_j)$ and then derive the key for ID using the key derivation algorithm. To derive the key for identity ID', $\mathcal{B}$ chooses the values $r_1, \ldots, r_j \xleftarrow{\$} \mathbb{Z}_p$ at random and sets $d_{ID'} = (a_0, a_1, \ldots, a_j)$ where

$$\begin{aligned} a_0 &\leftarrow g_2^{\frac{-\alpha_{j,0}}{\alpha_{j,1}(ID_j - ID_j^*)}} \cdot \prod_{i=1}^{j} \left(u_{i,0} \cdot u_{i,1}^{ID_i}\right)^{r_i} \\ a_i &\leftarrow g_1^{r_i} \qquad \text{for } i = 1, \ldots, j-1 \\ a_j &\leftarrow g_2^{\frac{-1}{\alpha_{j,1}(ID_j - ID_j^*)}} \cdot g_1^{r_j} \end{aligned}$$

Algorithm $\mathcal{A}_1$ terminates and outputs two equal-length messages (m_0, m_1).

Challenge. Algorithm $\mathcal{B}$ then chooses a random bit $\beta \xleftarrow{\$} \{0, 1\}$ and sends $C^* = (C, C^{\alpha_{1,0}}, \ldots, C^{\alpha_{\ell^*,0}}, m_\beta \cdot Z)$ to $\mathcal{A}$ as the challenge ciphertext. Since $u_{i,0} \cdot u_{i,1}^{ID_i^*} = g_1^{\alpha_{i,0}}$ for all i, we have

that

$$C^* = (g_1^c, (u_{1,0} \cdot u_{1,1}^{ID_i^*})^c, \ldots, (u_{\ell^*,0} \cdot u_{\ell^*,1}^{ID_{\ell^*}^*})^c, m_\beta \cdot Z).$$

As a result, when $Z = \hat{e}(g,g)^{abc} = \hat{e}(h_1, g_2)^c$, C^* is a valid encryption of message m_β for the challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$. On the other hand, when $Z = \hat{e}(g,g)^z$ for a random value $z \xleftarrow{\$} \mathbb{Z}_p$, then the challenge ciphertext is independent of β from the view point of the adversary.

Phase 2. $\mathcal{B}$ runs $\mathcal{A}_2$ on the challenge ciphertext C^*. If $\mathcal{A}_2$ makes any key derivation oracle queries, then they are answered as in Phase 1. $\mathcal{A}_2$ terminates and outputs a bit β'.

Output. If $\beta = \beta'$ then $\mathcal{B}$ outputs 1, guessing that $Z = \hat{e}(g,g)^{abc}$, otherwise $\mathcal{B}$ outputs 1.

Suppose E does not occur. Clearly, when $Z = \hat{e}(g,g)^{abc}$, the view of $\mathcal{A}$ is identical to its view in a real attack and, thus, the probability that $b = b'$ is exactly the probability that $\mathcal{A}$ wins the IND-sHID-CPA game. On the other hand, when Z is a random group element in $\mathbb{G}_T$, then the probability that $b = b'$ is exactly $1/2$. Hence, if E does not occur then $\mathcal{A}$ wins with probability ϵ. If E does occur, then the simulator fails; however, for E to occur then $\mathcal{A}$ must submit a key extraction query for an identity ID where ID^* is a prefix of ID and ID is a prefix of $\tilde{ID}^*$. This implies that $ID_{\ell^*+1} = ID_{\ell^*+1}^*$ but, since $ID_{\ell^*+1}^*$ is chosen at random and hidden from the execution of the attacker $\mathcal{A}$, we have that $\Pr[E] \leq q_K/p$. From the above, the result announced in Theorem E.2.7 follows immediately. ∎

E.2.7 The Boneh-Boyen-Goh Scheme

In this section we present the HIBE scheme due to Boneh, Boyen and Goh [BBG05], referred to as the BBG-HIBE scheme here. Again, we assume that identities are vectors of elements of $\mathbb{Z}_p$. The scheme is described in Figure E.2.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_i \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$
$mpk \leftarrow (g_1, g_2, h_1, u_0, \ldots, u_L)$
$d_0 \leftarrow h_2$
For $i = 1, \ldots, L+1$ do
  $d_i \leftarrow 1$
$msk \leftarrow (d_0, d_1, \ldots, d_L, d_{L+1})$
Return (mpk, msk)

Algorithm $\mathsf{KeyDer}(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_L, d_{L+1})$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d_0' \leftarrow d_0 \cdot d_{\ell+1}^{ID_{\ell+1}} \cdot \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^{r_{\ell+1}}$
For $i = \ell+2, \ldots, L$ do
  $d_i' \leftarrow d_i \cdot u_i^{r_{\ell+1}}$
$d_{L+1}' \leftarrow d_{L+1} \cdot g_1^{r_{\ell+1}}$
Return $(d_0', d_{\ell+2}', \ldots, d_L', d_{L+1}')$

Algorithm $\mathsf{Encrypt}(mpk, ID, m)$:
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
$C_2 \leftarrow \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return (C_1, C_2, C_3)

Algorithm $\mathsf{Decrypt}(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_{L+1})$
Parse C as (C_1, C_2, C_3)
$m' \leftarrow C_3 \cdot \frac{\hat{e}(C_2, d_{L+1})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.2: The Boneh-Boyen-Goh HIBE scheme.

The following theorem about the security of the scheme was proved in (the full version of) [BBG05].

Theorem E.2.8 If there exists a (t, q_K, ϵ)-adversary against the IND-sHID-CPA security of the BBG-HIBE (with hierarchy depth L) then there exists a (t', ϵ')-adversary against the L-BDHI problem in $\mathbb{G}$, where $\epsilon' \geq \epsilon$ and $t' \leq t + O(L \cdot q_K \cdot t_{exp})$ and t_{exp} is the time for an exponentiation in $\mathbb{G}$.

E.2.8 The Waters Scheme

Waters [Wat05] argued that his IBE scheme can easily be modified into an L-level HIBE scheme as per [BB04a]. Here we explicitly present this construction, that we refer to as the Waters-HIBE scheme. The scheme makes use of n-bit identities and is described in Figure E.3. The scheme makes use of group elements $(u_{1,0}, \ldots, u_{L,n})$ which are available as part of the scheme's public parameters. These group elements define a series of hash functions $(F_1, \ldots, F_L)$ where

$$F_i(ID_i) = u_{i,0} \prod_{j \in [ID_i]} u_{i,j} \, .$$

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L; j = 0 \ldots n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, ID, m):
Parse ID as $(ID_1, \ldots, ID_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1 \ldots \ell$ do
$C_{2,i} \leftarrow F_i(ID_i)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.3: The Waters HIBE scheme.

Waters [Wat05] informally states that the above HIBE scheme is IND-HID-CPA secure under the BDDH assumption, in the sense that if there exists a (t, q_K, ϵ)-adversary against the HIBE, then there exists an algorithm solving the BDDH problem with advantage $\epsilon' = O((n \cdot q_K)^L \epsilon)$. We shall assume in what follows that the Waters HIBE scheme is indeed IND-HID-CPA secure. However, the reader should be aware that any security results we state for schemes derived from the Water HIBE scheme are conjectural relative to the above assumption.

E.2.9 Hierarchical Identity-Based Key Encapsulation

One efficient paradigm for producing HIBE schemes is to the hybrid KEM-DEM construction. In the public key setting, this was first formally investigated by Cramer and Shoup [CS03] and extended to the identity-based setting by Bentahar *et al.* [BFMLS08]. A hybrid construction consists of an asymmetric KEM and a symmetric DEM.

A hierarchical identity-based KEM (HIB-KEM) consists of four algorithms ($\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap}$). The setup algorithm Setup and key derivation algorithm KeyDer have the same syntax as for a HIBE scheme. The encapsulation algorithm Encap takes as input a master public key mpk and an identity $ID = (ID_1, \ldots, ID_\ell)$ with $0 \leq \ell \leq L$; it outputs a symmetric key $K \in \{0,1\}^\lambda$ and an encapsulation C. The decapsulation algorithm Decap takes as input a private key d_{ID} and an encapsulation C, and outputs either a symmetric key $K \in \{0,1\}^\lambda$ or the error symbol $\perp$.

The security models for a HIB-KEM is similar to those of a HIBE scheme. The IND-HID-CCA game for a HIB-KEM, played between an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger, is defined as follows:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that will, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$ and a ciphertext C, return $\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs a challenge identity $ID^* = (ID_1^*, \ldots, ID_{\ell^*}^*)$ and some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$, computes the encapsulation $(C^*, K_0) \xleftarrow{\$} \mathsf{Encap}(mpk, ID^*)$ and chooses a random key $K_1 \xleftarrow{\$} \{0,1\}^\lambda$.
4. The adversary runs $\mathcal{A}_2$ on the input (C^*, K_β) and the state information *state*. The adversary is given access to a key derivation oracle and decryption oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle with any ancestor identity of ID^*, and if it doesn't query the decryption oracle on the pair (ID^*, C^*) after it receives the challenge ciphertext. As usual, the adversary's advantage is defined to be equal to $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition E.2.9 A (t, q_K, q_D, ϵ)-adversary against the IND-HID-CCA security of the HIB-KEM is an algorithm that runs in time at most t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-HID-CCA game described above.

Again, if the random oracle model [BR93] is used in the analysis of a scheme, then the above security definitions take an extra parameter q_H as input. This parameter denotes the adversary's maximum number of queries to the random oracle.

A DEM is a pair of deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$. The encryption algorithm Enc takes as input a symmetric key $K \in \{0,1\}^\lambda$ and a message m of arbitrary length, and outputs a ciphertext $C \leftarrow \mathsf{Dec}(K, C)$. The decryption algorithm Dec takes as input a symmetric key $K \in \{0,1\}^\lambda$ and a ciphertext C, and returns either a message m or the error symbol $\perp$. The DEM must satisfy the following soundness property: for all $K \in \{0,1\}^\lambda$ and for all $m \in \{0,1\}^*$, we have that $\mathsf{Dec}(K, \mathsf{Enc}(K, m)) = m$.

The only security model which will concern us for DEMs is the (one-time) IND-CCA security game, which is played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a key $K \xleftarrow{\$} \{0,1\}^\lambda$.
2. The adversary runs $\mathcal{A}_1$. The adversary outputs two equal-length messages (m_0, m_1) and some state information *state*.

3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \leftarrow \mathsf{Enc}(K, m_\beta)$.

4. The adversary runs $\mathcal{A}_2$ on input C^* and the state information *state*. The adversary may query a decryption oracle which will, on input of a ciphertext $C \neq C^*$, return $\mathsf{Dec}(K, C)$. The adversary outputs a bit β'.

The adversary wins if $\beta = \beta'$ and its advantage is defined to be $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition E.2.10 A (t, q_D, ϵ)-adversary against the (one-time) IND-CCA security of the DEM is an algorithm that runs in time at most t, makes at most q_D decryption oracle queries, and has advantage at least ϵ in winning the IND-CCA game described above.

A HIB-KEM and a DEM can be "glued" together to form a complete HIBE scheme. Further details can be found in [BFMLS08].

E.2.10 The Canetti-Halevi-Katz Transform

We shall, in one of our constructions of a CCA WIBE scheme, make use of the techniques behind the Canetti-Halevi-Katz transform [CHK04]. To aid the reader we recap on this here. This is a transform to turn a weakly secure (IND-sID-CPA) IBE scheme into a fully secure (IND-CCA) public key encryption scheme. We let $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ denote the key-generation, extraction, encryption, and decryption algorithms of the IBE scheme, and $(\mathsf{Setup}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ denote the key-generation, encryption, and decryption algorithms of the derived public key scheme. The transform also makes use of a one-time signature scheme, defined by a tuple of algorithms $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$.

The algorithm Setup' is defined to be equal to Setup, i.e. public/private key of the PKE scheme is the master public/private keys, (mpk, msk), of the IBE scheme. Algorithm $\mathsf{Encrypt}'$ is defined as follows: First a key-pair (sk, vk) for the one-time signature scheme is created by calling SigGen; then the message is encrypted via $\mathsf{Encrypt}(mpk, vk, m)$ with respect to the "identity" vk to produce c. The resulting ciphertext c is then signed with sk to produce $\sigma = \mathsf{Sign}(sk, c)$. The tuple (vk, c, σ) is the ciphertext for our PKE.

To decrypt the recipient first verifies σ is a valid signature on c with respect to the verification key vk, by calling $\mathsf{Verify}(vk, c, \sigma)$. If it is then the function KeyDer is called with respect to the "identity" vk, using private key of the PKE (i.e. msk). Then the ciphertext can be decrypted using the algorithm $\mathsf{Decrypt}$.

E.3 Wildcard Identity-Based Encryption

E.3.1 Syntax

Identity-based encryption with wildcards (WIBE) schemes are essentially a generalisation of HIBE schemes where at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. Such a pattern is described by a vector $P = (P_1, \ldots, P_\ell) \in (\{0,1\}^* \cup \{*\})^\ell$, where $*$ is a special wildcard symbol. We say that identity $ID = (ID_1, \ldots, ID_{\ell'})$ *matches* P, denoted $ID \in_* P$, if and only if $\ell' \leq \ell$ and for all $i = 1, \ldots, \ell'$ we have that $ID_i = P_i$ or $P_i = *$. Note that under this definition, any ancestor of a matching identity is also a matching identity. This is reasonable for our purposes because any ancestor can derive the secret key of a matching descendant identity anyway.

If $P = (P_1, \ldots, P_\ell)$ is a pattern, then we define $W(P)$ to be the set of wildcard positions in P, i.e.

$$W(P) = \{1 \leq i \leq \ell \ : \ P_i = *\}.$$

Formally, a WIBE scheme is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ providing the following functionality. The Setup and KeyDer algorithms behave exactly as those of a HIBE scheme. To create a ciphertext of a message $m \in \{0,1\}^*$ intended for all identities matching pattern P, the sender computes $C \xleftarrow{\$} \mathsf{Encrypt}(mpk, P, m)$. Any of the intended recipients $ID \in_* P$ can decrypt the ciphertext using its own decryption key as $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$.

Note that we implicitly assume that the pattern P used to encrypt the message is included within the ciphertext. This is because any parent of the pattern should be able to decrypt the message, and hence the parent will need to be able to fill in the non-wildcarded entries in the pattern for decryption. For example, suppose the pattern is $P = (ID_1, *, ID_3)$ and that the decryptor has identity $ID = (ID_1, ID_2)$. Then by our definition of a matching pattern we have $ID \in_* P$, and so the decryptor will need to be informed of ID_3 so as to be able to decrypt the ciphertext. Note that an anonymous version of the definitions can be presented, but we do not consider this further in this paper for simplicity.

Correctness requires that for all key pairs (mpk, msk) output by Setup, all messages $m \in \{0,1\}^*$, all $0 \leq \ell \leq L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^\ell$, and all identities $ID \in_* P$, we have

$$\mathsf{Decrypt}(\ \mathsf{KeyDer}(msk, ID)\ ,\ \mathsf{Encrypt}(mpk, P, m)\ ) = m$$

with probability one.

E.3.2 Security Notions

We define the security of WIBE schemes analogously to that of HIBE schemes, but with the adversary choosing a challenge pattern instead of an identity to which the challenge ciphertext will be encrypted. To exclude trivial attacks, the adversary is not able to query the key derivation oracle on any identity that matches the challenge pattern, nor is it able to query the decryption oracle on the challenge ciphertext in combination with any identity matching the challenge pattern.

More formally, the IND-WID-CPA security model is defined through the following game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.

2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary outputs two equal-length messages (m_0, m_1) and a challenge pattern P^*, along with some state information *state*.

3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m_\beta)$.

4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$ and it never queries the decryption oracle on any identity ID which matches the pattern P^*, i.e. any identity $ID \in_* P^*$. The adversary's advantage is defined as $|2 \cdot \Pr[\,\mathcal{A} \text{ wins}\,] - 1|$.

Definition E.3.1 A (t, q_K, ϵ)-adversary against the IND-WID-CPA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, and has advantage at least ϵ in the IND-WID-CPA game described above.

In the IND-WID-CCA security model is identical to the IND-WID-CPA security model with the exception that the adversary has access to a decryption oracle, which will, on input of an identity ID and a ciphertext C, return $\mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary wins the game if $\beta = \beta'$, it never queries the decryption oracle on any identity $ID \in_* P^*$, and the adversary doesn't query the decryption oracle the combination of any identity $ID \in_* P^*$ and the ciphertext C^*. The adversary's advantage is defined as $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition E.3.2 A (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, makes at most q_D decryption oracle queries, and has advantage at least ϵ in the IND-WID-CCA game described above.

As for the case of HIBEs, we also define a weaker selective-identity (sWID) security notion, in which the adversary commits to the challenge pattern at the beginning of the game, before the master public key is made available. The notions of IND-sWID-CPA and IND-sWID-CCA security are defined analogously to the above. In the random oracle model, the additional parameter q_H denotes the adversary's maximum number of queries to the random oracle, or the total number of queries to all random oracles when it has access to multiple ones.

If the WIBE scheme has a finite message space $\mathcal{M}$, then we may also define a one-way notion for encryption security (OW-WID-CPA). This is formally defined via the following game, played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on input mpk. The adversary is given access to a key derivation oracle as in the IND-WID-CPA game. The adversary outputs a challenge pattern P^* and some state information $state$.
3. The challenger generates $m \xleftarrow{\$} \mathcal{M}$ and computes the ciphertext $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information $state$. The adversary is given access to a key derivation oracle as before. It output a message m'.

The adversary wins the game if $m = m'$ and the adversary never queries the key derivation oracle on an identity $ID \in_* P^*$. The adversary's advantage is defined to be $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Definition E.3.3 A (t, q_K, ϵ)-adversary against the OW-WID-CPA security of the WIBE scheme is an algorithm that runs in time at most t, makes at most q_K key derivation oracle queries, and has advantage at least ϵ in winning the OW-WID-CPA game described above.

E.3.3 Constructing a WIBE from a HIBE

In order to clarify the relationship between HIBEs and WIBEs, we first point out a generic construction of a WIBE scheme from any HIBE scheme. However, this WIBE scheme has a secret key size that is exponential in the depth of the hierarchy tree. Let "$*$" denote a dedicated bitstring that cannot occur as a user identity. Then the secret key of a user with identity $(ID_1, \ldots, ID_\ell)$ in the WIBE scheme contains the 2^ℓ HIBE secret keys of all patterns matching this identity. For example, the secret key of identity (ID_1, ID_2) contains four HIBE secret keys, namely those corresponding to identities

$$(ID_1, ID_2), (\text{“}*\text{”}, ID_2), (ID_1, \text{“}*\text{”}), (\text{“}*\text{”}, \text{“}*\text{”}).$$

To encrypt to a pattern $(P_1, \ldots, P_\ell)$, one uses the HIBE scheme to encrypt to the identity obtained by replacing each wildcard in the pattern with the "$*$" string, i.e. the identity $(ID_1, \ldots, ID_\ell)$ where $ID_i =$ "$*$" if $P_i = *$ and $ID_i = P_i$ otherwise. The final WIBE ciphertext consists of the pattern and the HIBE ciphertext. Decryption is done by selecting the appropriate secret key from the list and using the decryption algorithm of the HIBE scheme.

Theorem E.3.4 If there exists a (t, q_K, ϵ) attacker against the IND-WID-CPA security of the WIBE scheme (with hierarchy depth L) then there exists a $(t', 2^L q_K, \epsilon)$-adversary against the IND-HID-CPA security of the corresponding HIBE scheme, where $t' \leq t + 2^L q_K t_K$ and t_K is the time taken to compute a key derivation query. If there exists a (t, q_K, q_D, ϵ) attacker against the IND-WID-CCA security of the WIBE scheme (with hierarchy depth L) then there exists a $(t', 2^L q_K, q_D, \epsilon)$-adversary against the IND-HID-CCA security of the corresponding HIBE scheme, where $t' \leq t + 2^L q_K t_K + q_D t_D$, t_K is the time taken to compute a key derivation query, and t_D is the time taken to compute a decryption query.

Notice that the appearance of the term 2^L in the security reduction means that this construction is only guaranteed to be secure when the number of levels grows poly-logarithmically in the secure parameter. This restriction occurs in the security analysis of all the HIBE schemes that we consider.

The efficiency of the WIBE scheme obtained with this construction is roughly the same as that of the underlying HIBE scheme, but with the major disadvantage that the size of the secret key is 2^ℓ times that of a secret key in the underlying HIBE scheme. This is highly undesirable for many applications, especially since the secret key may very well be kept on an expensive secure storage device. It is interesting to investigate whether WIBE schemes exist with overhead polynomial in all parameters. We answer this question in the affirmative here by presenting direct schemes with secret key size linear in ℓ. Unfortunately, for all of our schemes, this reduction in key size comes at the cost of linear-size ciphertexts, while the generic scheme can achieve constant-size ciphertexts when underlain by a HIBE with constant ciphertext size, e.g. that of [BBG05].

E.3.4 The Relationship Between WIBE and GIBE, FIBE, and ABE

As we have seen WIBEs are closely related to HIBEs. They are also related to a concept called Generalised Identity-Based Encryption (GIBE) [BH08]. In a GIBE one has a set of policies P and a set of roles R. The roles are partially ordered so that a "higher" role can delegate its abilities to a "lower" role. Whether a party can decrypt a ciphertext depends on whether a predicate defined on the set $P \times R$ evaluates to true. In particular a ciphertext is encrypted to a policy $\pi \in P$, and it can be decrypted by a role ρ if and only if the predicate evaluated on (π, ρ) evaluates to true. It is easy to see that the roles in a GIBE correspond to the identities in a WIBE, whilst the policies correspond to the wildcarded patterns. Hence, a WIBE is a specific example of a GIBE. However, the expressive nature of a GIBE being greater than that of a WIBE comes at a cost, in that one can construct WIBE schemes which are more secure than the equivalent GIBE.

Another related primitive is fuzzy identity-based encryption (FIBE) [SW05], which allows a ciphertext encrypted to identity ID to be decrypted by any identity ID' that is "close" to ID according to some metric. In the schemes of [SW05], an identity is a subset containing n elements from a finite universe. Two identities ID and ID' are considered "close" if $|ID \cap ID'| \geq d$ for some parameter d. A FIBE with $n = 2L$ and $d = L$ can be used to construct a WIBE scheme (without hierarchical key derivation) by letting the decryption key for identity $(ID_1, \ldots, ID_\ell)$

correspond to the decryption key for the set

$$\{1\|ID_1, \ldots, \ell\|ID_\ell,\ \ (\ell+1)\|\varepsilon, \ldots, L\|\varepsilon,\\ 1\|\text{“}*\text{”}, \ldots, L\|\text{“}*\text{”}\}\ .$$

Suppose that “$\perp$” is a unique string which cannot occur as a user identity and distinct from “$*$”. One can encrypt to pattern $P = (P_1, \ldots, P_\ell)$ by encrypting to the set

$$\{1\|P'_1, \ldots, \ell\|P'_\ell,\ (\ell+1)\|\varepsilon, \ldots, L\|\varepsilon, 1\|\text{“}\perp\text{”}, \ldots, L\|\text{“}\perp\text{”}\},$$

where the $P'_i \leftarrow P_i$ if $i \notin W(P)$ and $P'_i \leftarrow$ “$*$” if $i \in W(P)$. The dummy symbols “$\perp$” are only used to ensure that the size of the encryption set is exactly $2L$ (as required by the definition of the FIBE scheme). We stress that this construction does not give a full WIBE scheme as it does not permit hierarchical key derivation. This also implies that a “parent” identity cannot decrypt message sent to its “children” identities as it cannot derive the key for the child.

Fuzzy-IBE, GIBEs, and WIBEs are themselves examples of a policy-based encryption mechanisms. In such systems access to encrypted data is provided as long as the recipient has a key (or set of keys) which correspond to some policy. The power of identity-based mechanisms to enable policy-based access control to encrypted data was realised very early on in the history of pairing-based IBE [Sma03]. In recent years this idea has been formalised under the heading of Attribute Based Encryption.

In Attribute Based Encryption [SW05], or, more correctly, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [BSW07, GPSW06], a recipient is issued keys corresponding to a number of credentials. An encryptor will encrypt a message under a policy, i.e. a set of credentials which are required by any user who wishes to obtain access to the message. Any recipient which has credential key which meet the policy statement has access to the encrypted data. The defining characteristic of CP-ABE is that the policies are embedded in the ciphertexts.

In the context of WIBEs the policy is that the user should have a key (credential) which matches the pattern. For a pattern such as $(ID_1, *, ID_3)$ this can be interpreted as having a credential for an identity with ID_1 in the first position and an identity with ID_3 in the third position. However, a CP-ABE scheme would offer separate credentials (keys) for each position, whereas a WIBE compresses all of these credentials in a single key. Hence, ABE is clearly a more powerful concept than a WIBE, as it allows more expressive policies, but WIBE schemes are often simpler to construct.

E.4 Identity-based Key Encapsulation with Wildcards

We can also define a notion of Identity-Based Key Encapsulation Mechanism with Wildcards (WIB-KEM). A WIB-KEM consists of the following four algorithms ($\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}$, Decap). The algorithms Setup and KeyDer are defined as in the WIBE case. The encapsulation algorithm Encap takes the master public key mpk of the system and a pattern P, and returns (C, K), where $K \in \{0,1\}^\lambda$ is a symmetric key and C is an encapsulation of the key K. Again we assume that the encapsulation includes a public encoding of the pattern P under which the message has been encrypted. Finally, the decapsulation algorithm $\mathsf{Decap}(mpk, d_{ID}, C)$ takes a private key d_{ID} and an encapsulation C, and returns either a secret key K or the error symbol $\perp$.

A WIB-KEM must satisfy the following soundness property: for all pairs (mpk, msk) output by Setup, all $0 \le \ell \le L$, all patterns $P \in (\{0,1\}^* \cup \{*\})^\ell$, and all identities $ID \in_* P$, we have

$$\Pr\left[K' = K : (C, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P); K' \xleftarrow{\$} \mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)\right] = 1\,.$$

The IND-WID game for WIB-KEMs is similar to both the IND-WIB game for WIBEs and the IND-HIB game for HIB-KEMs. The IND-WIB-CCA game is played between an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a challenger:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that will, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$ and a ciphertext C, return $\mathsf{Decap}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs a challenge pattern P^* and some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$, computes the encapsulation $(C^*, K_0) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and chooses a random key $K_1 \xleftarrow{\$} \{0,1\}^\lambda$.
4. The adversary runs $\mathcal{A}_2$ on the input (C^*, K_β) and the state information *state*. The adversary is given access to a key derivation oracle and decryption oracle as before. The adversary outputs a bit β'.

The adversary wins the game if $\beta = \beta'$, it never queries the key derivation oracle on any identity $ID \in_* P^*$, and if it doesn't query the decryption oracle on the pair (ID, C^*) for some $ID \in_* P^*$ after it receives the challenge ciphertext. As usual, the adversary's advantage is defined to be equal to $|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1|$.

Another common form for writing the advantage of an IND-WID-CCA adversary for a WIB-KEM is given by the following simple lemma.

Lemma E.4.1 If $\mathcal{A}$ is a (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIB-KEM and β, β' are as in the IND-WID-CCA security game, then

$$\epsilon = |\Pr[\beta' = 1 \mid \beta = 1] - \Pr[\beta' = 1 \mid \beta = 0]|$$

Definition E.4.2 A (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of a HIB-KEM is an algorithm that runs in time t, makes at most q_K queries to the key derivation oracle, makes at most q_D queries to the decryption oracle, and has advantage at least ϵ in winning the IND-WID-CCA game described above.

We may combine a WIB-KEM $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$ with a DEM $(\mathsf{Enc}, \mathsf{Dec})$ (see Section E.2.9) to form a complete WIBE scheme $(\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$, where the encryption and decryption algorithms are as follows:

- $\mathsf{Encrypt}(mpk, P^*, m)$:
  1. Compute $(C_1, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$.
  2. Compute $C_2 \leftarrow \mathsf{Enc}(K, m)$.
  3. Output the ciphertext $C = (C_1, C_2)$.
- $\mathsf{Decrypt}(d_{ID}, C)$:
  1. Parse C as (C_1, C_2).
  2. Compute $K \xleftarrow{\$} \mathsf{Decap}(d_{ID}, C_1)$. If $K = \perp$ then output $\perp$.
  3. Compute $m \leftarrow \mathsf{Dec}(K, C_2)$.

4. Output m.

Theorem E.4.3 If there exists a (t, q_K, q_D, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against IND-WID-CCA security of the hybrid WIBE, then there is a $(t_{\mathcal{B}}, q_K, q_D, \epsilon_{\mathcal{B}})$-adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM and a $(t_{\mathcal{B}'}, q_D, \epsilon_{\mathcal{B}'})$-adversary $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ against the IND-CCA security of the DEM such that:

$$\begin{aligned} t_{\mathcal{B}} &\leq t + q_D t_{\mathsf{Dec}} + t_{\mathsf{Enc}} \\ t_{\mathcal{B}'} &\leq t + q_D(t_{\mathsf{Dec}} + t_{\mathsf{Decap}} + t_{\mathsf{KeyDer}}) + q_K t_{\mathsf{KeyDer}} \\ &\quad + t_{\mathsf{Encap}} + t_{\mathsf{Setup}} \\ \epsilon &\leq 2\epsilon_{\mathcal{B}'} + \epsilon_{\mathcal{B}} \end{aligned}$$

where t_{Enc} is the time to run the DEM's Enc algorithm, t_{Dec} is the time to run the DEM's Dec algorithm, t_{Setup} is the time to run the KEM's Setup algorithm, t_{Decap} is the time to run the KEM's Decap algorithm and t_{KeyDer} is the time to run the KEM's KeyDer algorithm.

Proof: This proof mirrors the proofs of Cramer and Shoup [CS03] and Bentahar *et al.* [BFMLS08]. We prove this result in two stages. First, we change the nature of the security game. Let Game 1 be the normal IND-WID-CCA game for the WIBE scheme. Let Game 2 be the slight adaptation of the IND-WID-CCA game:

1. The challenger generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
2. The adversary runs $\mathcal{A}_1$ on mpk. The adversary is given access to a key derivation oracle that, on input of an identity $ID = (ID_1, \ldots, ID_\ell)$, returns the secret key $d_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$ corresponding to that identity. The adversary is also given access to a decryption oracle that, on input of an identity ID and a ciphertext C, returns $\mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C)$. The adversary outputs two messages (m_0, m_1) of equal length and a challenge pattern P^*, along with some state information *state*.
3. The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and a key $K^* \xleftarrow{\$} \{0,1\}^\lambda$, then computes the ciphertext $(C_1^*, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and $C_2 \leftarrow \mathsf{Enc}(K^*, m_\beta)$. The challenge ciphertext is $C^* \leftarrow (C_1^*, C_2^*)$.
4. The adversary runs $\mathcal{A}_2$ on the input C^* and the state information *state*. The adversary is given access to a key derivation oracle as before. The adversary is also given to a decryption oracle that, on input of an identity ID and a ciphertext $C = (C_1, C_2)$, returns

$$\begin{cases} \mathsf{Decrypt}(\mathsf{KeyDer}(msk, ID), C) & \text{if } ID \not\in_* P^* \text{ or } C_1 \neq C_1^* \\ \mathsf{Dec}(K^*, C_2) & \text{if } ID \in_* P^* \text{ and } C_1 = C_1^*) \end{cases}.$$

   The adversary outputs a bit β'.

Note that the only two differences between the game and the IND-WID-CCA game are that a random key is used to compute the challenge ciphertext and to decrypt certain ciphertexts after the challenge ciphertext is issued.

We show that any change in the actions of $\mathcal{A}$ between Game 1 and Game 2 give rise to an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-WID-CCA security of the WIB-KEM. We describe the algorithm $\mathcal{B}_1$ below:

1. $\mathcal{B}_1$ takes as input the master public key mpk.
2. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle query, then $\mathcal{B}_1$ forwards this query to its own oracle and returns the result. If $\mathcal{A}_1$ makes a decryption oracle query on an identity ID and a ciphertext (C_1, C_2), the $\mathcal{B}_1$ forwards C_1 to its decapsulation oracle and receives a key K in return. $\mathcal{B}_1$ returns $\mathsf{Dec}(K, C_2)$ to $\mathcal{A}$. $\mathcal{A}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
3. $\mathcal{B}_1$ outputs the challenge pattern P^*.

The challenger then computes a challenge encapsulation (C_1^*, K^*) where K^* is either the decapsulation of C_1^* or a random key. The algorithm $\mathcal{B}_2$ runs as follows:

1. $\mathcal{B}_2$ takes as input the challenge encapsulation (C_1^*, K^*). $\mathcal{B}_2$ chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes the remainder of the challenge ciphertext $C_2^* \leftarrow \mathsf{Enc}(K^*, m_\beta)$.
2. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the challenge ciphertext $C^* = (C_1^*, C_2^*)$. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}_2$ forwards this query to its own oracle and returns the result. If $\mathcal{A}_2$ makes a decryption oracle query on an identity $ID \in_* P^*$ and a ciphertext (C_1^*, C_2), then $\mathcal{B}_2$ returns $\mathsf{Dec}(K^*, C_2)$ to $\mathcal{A}_2$. Otherwise, if $\mathcal{A}_2$ makes a decryption oracle query on an identity ID and a ciphertext (C_1, C_2), then $\mathcal{B}_2$ answers the query as before, by querying its own oracle to find the decapsulation of C_1 and decrypting C_2 itself. $\mathcal{A}_2$ outputs a bit β'.
3. If $\beta = \beta'$ then $\mathcal{B}_2$ outputs 1; otherwise $\mathcal{B}_2$ outputs 0.

If K^* is the decapsulation of C_1^* then $\mathcal{B}$ simulates Game 1 for $\mathcal{A}$; whereas if K^* is a random key then $\mathcal{B}$ simulates Game 2 for $\mathcal{A}$. Thus we have,

$$|\Pr[\,\mathcal{A} \text{ wins in Game } 1\,] - \Pr[\,\mathcal{A} \text{ wins in Game } 2\,]| = \epsilon_{\mathcal{B}}$$

by virtue of Lemma E.4.1.

However, the security of Game 2 depends only on the (one-time) IND-CCA security of the DEM. We give an algorithm $\mathcal{B}' = (\mathcal{B}_1', \mathcal{B}_2')$ reduces the security of the WIBE in Game 2 to the security of the DEM. We describe the algorithm $\mathcal{B}_1'$ below:

1. $\mathcal{B}_1'$ computes $(mpk, msk) \leftarrow \mathsf{Setup}$.
2. $\mathcal{B}_1'$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation or decryption oracle query, then $\mathcal{B}_1'$ computes the correct answer using its knowledge of the master private key msk. $\mathcal{A}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
3. $\mathcal{B}_1'$ outputs the messages (m_0, m_1).

The challenger chooses a bit $\beta \xleftarrow{\$} \{0,1\}$ and computes $C_2^* \xleftarrow{\$} \mathsf{Enc}(K^*, m_\beta)$ as the challenge encryption using a randomly chosen (and hidden) key $K^* \xleftarrow{\$} \{0,1\}^\lambda$. The algorithm $\mathcal{B}_2'$ runs as follows:

1. $\mathcal{B}_2'$ takes C_2^* as input. $\mathcal{B}_2'$ computes the encapsulation $(C_1^*, K) \xleftarrow{\$} \mathsf{Encap}(mpk, P^*)$ and sets the challenge ciphertext $C^* \leftarrow (C_1^*, C_2^*)$.

2. $\mathcal{B}'_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}'_2$ answers it correctly using its knowledge of the master private key msk. If $\mathcal{A}_2$ makes a decryption oracle query on an identity $ID \in_* P^*$ and a ciphertext (C_1^*, C_2) then $\mathcal{B}'_2$ computes the correct answer by querying its own decryption on C_2 and returning the result. Otherwise, if $\mathcal{A}_2$ makes a decryption oracle query on an identity ID and a ciphertext C, then $\mathcal{B}'_2$ computes the correct answer using its knowledge of the master private key msk. $\mathcal{A}_2$ outputs a bit β'.

3. $\mathcal{B}'_2$ outputs the bit β'.

$\mathcal{B}'$ correctly simulates Game 2 for $\mathcal{A}$. Furthermore, $\mathcal{A}$ wins in Game 2 if and only if $\mathcal{B}$ wins the IND-CCA game for a DEM. Hence,

$$|2 \cdot \Pr[\,\mathcal{A} \text{ wins Game } 2\,] - 1| = \epsilon_{\mathcal{B}'}$$

and so we have that

$$\begin{aligned} \epsilon &= |2 \cdot \Pr[\,\mathcal{A} \text{ wins Game } 1\,] - 1| \\ &\leq 2 \cdot |\Pr[\,\mathcal{A} \text{ wins in Game } 1\,] - \Pr[\,\mathcal{A} \text{ wins in Game } 2\,]| \\ &\qquad + |2 \cdot \Pr[\,\mathcal{A} \text{ wins in Game } 2\,] - 1| \\ &= 2\epsilon_{\mathcal{B}} + \epsilon_{\mathcal{B}'}\,. \end{aligned}$$

∎

E.5 IND-WID-CPA Secure WIBEs

In this section, we propose several WIBE schemes which are IND-WID-CPA secure, based on three existing HIBE schemes from the Boneh-Boyen family (BB-HIBE, BBG-HIBE, Waters-HIBE). These three direct constructions all utilize a similar technique of modifying a HIBE's ciphertext generation to include some extra data related to each wildcard. The security proof then reduces the security of the resulting WIBE to that of the underlying HIBE. These schemes are all proven secure using the same "projection" technique and so we only prove the security of one scheme (Waters-WIBE) relative to the security of the underlying HIBE (in this case Waters-HIBE). Note, in that due to our earlier comment on the lack of a full security proof for the Waters-HIBE, we obtain a full security theorem only for the cases of the BB- and BBG-based WIBE's.

Each of these three schemes is proven secure, relative to the underlying HIBE, in the standard model; however, two of these schemes are only proven secure in the IND-sWID-CPA model. We therefore give a generic transformation from an IND-sWID-CPA secure scheme to an IND-WID-CPA secure scheme which uses the random oracle model.

E.5.1 The Boneh-Boyen WIBE

Our first construction is based on the slight variant of the BB-HIBE [BB04a] which we prove secure in Section E.2.6. As with the BB-HIBE scheme, the BB-WIBE makes use of identities which are vectors of elements of $\mathbb{Z}_p$. The scheme is described in Figure E.4. Note that the decryption algorithm can determine if $i \in W(P)$ by checking whether $C_{2,i}$ contains one group element or two.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1 \ldots L, j = 0, 1$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,1})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot \left(u_{\ell+1,0} \cdot u_{\ell+1,1}^{ID_{\ell+1}}\right)^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C_{2,i} \leftarrow \left(u_{i,0} \cdot u_{i,1}^{P_i}\right)^r$
If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, u_{i,1}^r)$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_l)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
If $i \in W(P)$ then
Parse $C_{2,i}$ as (v_1, v_2)
$C'_{2,i} \leftarrow v_1 \cdot v_2^{ID_i}$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.4: The Boneh-Boyen WIBE scheme.

The BB-WIBE can actually be seen as a close relative of the Waters-WIBE scheme (see Section E.5.3) with the hash function $F_i(ID_i)$ being defined as

$$F_i(ID_i) = u_{i,0} \cdot u_{i,1}^{ID_i} .$$

Its security properties are different though since the BB-WIBE scheme can be proved secure in the selective-identity model only. We reduce its security to that of the BB-HIBE scheme, which in its turn is proved IND-sHID-CPA secure under the BDDH assumption in Section E.2.6. The proof of the theorem below is analogous to that of Theorem E.5.3, and hence omitted. One important difference with Theorem E.5.3 is that the reduction from the BB-HIBE scheme is tight: because we prove security in the selective-identity model, we do not lose a factor 2^L due to having to guess the challenge pattern upfront.

Theorem E.5.1 If there exists a (t, q_K, ϵ)-adversary against the IND-sWID-CPA security of a BB-WIBE (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-sHID-CPA security of the BB-HIBE, where

$$t' \leq t + 2L(1 + q_K) \cdot t_{exp} , \quad q'_K \leq q_K \quad \text{and } \epsilon' \geq \epsilon ,$$

where t_{exp} is the time required to compute an exponentiation in $\mathbb{G}$.

In terms of efficiency, the BB-WIBE scheme easily outperforms the Waters-WIBE scheme: the master public key contains $2L+3$ group elements. Encryption to a recipient pattern of length ℓ and w wildcards involves $\ell+w+2$ (multi-)exponentiations and produces ciphertexts containing $\ell+w+2$ group elements, or $2L+2$ group elements in the worst case that $\ell = w = L$. Decryption requires the computation of $\ell + 1$ pairings, just like the Waters-WIBE scheme. However, this scheme is outperformed by the BBG-WIBE.

E.5.2 The Boneh-Boyen-Goh WIBE

Our second construction is based on the BBG-HIBE [BBG05] (see Section E.2.7). The BBG-HIBE scheme has the advantage of constant-sized ciphertexts. Our BBG-WIBE scheme does not have this advantage, but does have the advantage that a pattern with w wildcards leads to a ciphertext with $w+3$ elements and is secure under the same decisional L-BDHI problem as the BBG-HIBE. Again, identities are considered to be vectors of elements of $\mathbb{Z}_p$ and the scheme is given in Figure E.5.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_i \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L$
$mpk \leftarrow (g_1, g_2, h_1, u_0, \ldots, u_L)$
$d_0 \leftarrow h_2$
For $i = 1, \ldots, L+1$ do
  $d_i \leftarrow 1$
$msk \leftarrow (d_0, d_1, \ldots, d_L, d_{L+1})$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_L, d_{L+1})$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot d_{\ell+1}^{ID_{\ell+1}} \cdot \left(u_0 \prod_{i=1}^{\ell} u_i^{ID_i}\right)^{r_{\ell+1}}$
For $i = \ell+2, \ldots, L$ do
  $d'_i \leftarrow d_i \cdot u_i^{r_{\ell+1}}$
$d'_{L+1} \leftarrow d_{L+1} \cdot g_1^{r_{\ell+1}}$
Return $(d'_0, d'_{\ell+2}, \ldots, d'_L, d'_{L+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
$C_2 \leftarrow \left(u_0 \prod_{i=1, i \notin W(P)}^{\ell} u_i^{P_i}\right)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
$C_4 \leftarrow (u_i^r)_{i \in W(P)}$
Return (P, C_1, C_2, C_3, C_4)

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, d_{\ell+1}, \ldots, d_{L+1})$
Parse C as (P, C_1, C_2, C_3, C_4)
Parse C_4 as $(v_i)_{i \in W(P)}$
$C'_2 \leftarrow C_2 \prod_{i=1, i \in W(P)}^{\ell} v_i^{ID_i}$
$m' \leftarrow C_3 \cdot \frac{\hat{e}(C'_2, d_{L+1})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.5: The Boneh-Boyen-Goh WIBE scheme.

The BBG-WIBE scheme is significantly more efficient than the Waters-WIBE and BB-WIBE schemes in terms of decryption, and also offers more efficient encryption and shorter ciphertexts when the recipient pattern contains few wildcards. More precisely, the master public key contains $L+4$ group elements. Encryption to a recipient pattern of length ℓ with w wildcards involves $w+3$ (multi-)exponentiations and $w+3$ group elements in the ciphertext, or $L+3$ of these in the worst case that $\ell = w = L$. Decryption requires the computation of two pairings, as opposed to $\ell+1$ of these for the Waters-WIBE and BB-WIBE schemes.

Again, the proof of the following theorem is analogous to that of Theorem E.5.3, and hence omitted.

Theorem E.5.2 If there is a (t, q_K, ϵ)-adversary against the IND-sWID-CPA security of the BBG-WIBE (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-sHID-CPA security of the BBG-HIBE where

$$t' \leq t - L(1 + 2q_K) \cdot t_{exp}, \quad q'_K \leq q_K, \text{ and } \epsilon' \geq \epsilon,$$

where t_{exp} is the time it takes to perform an exponentiation in $\mathbb{G}$.

E.5.3 The Waters WIBE

Our third construction is based on the Waters-HIBE [Wat05] (see Section E.2.8). As in the HIBE scheme, the WIBE makes use of identities which are n-bit strings and a series of hash

functions $(F_1, \ldots, F_L)$ where

$$F_i(ID_i) = u_{i,0} \prod_{j \in [ID_i]} u_{i,j} \,.$$

The scheme is described in Figure E.6.

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^\alpha$; $h_2 \leftarrow g_2^\alpha$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L; j = 0 \ldots n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm KeyDer$(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm Encrypt(mpk, P, m):
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1 \ldots \ell$ do
If $i \notin W(P)$ then $C_{2,i} \leftarrow F_i(ID_i)^r$
If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, \ldots, u_{i,n}^r)$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
Return $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$

Algorithm Decrypt$(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_3)$
For $i = 1, \ldots, \ell$ do
If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
If $i \in W(P)$ then
Parse $C_{2,i}$ as $(v_0, \ldots, v_n)$
$C'_{2,i} \leftarrow v_0 \prod_{i \in [ID_i]} v_i$
$m' \leftarrow C_3 \cdot \frac{\prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, d_0)}$
Return m'

Figure E.6: The Waters WIBE scheme.

In terms of efficiency, the Waters-WIBE compares unfavourably with the BB-WIBE and BBG-WIBE (but (conjecturally) provides stronger security guarantees in the standard model). The master public key of the Waters-WIBE scheme contains $(n + 1)L + 3$ group elements. Encrypting to a pattern of length ℓ containing w wildcards comes at the cost of $\ell + nw + 2$ exponentiations and $\ell + nw + 2$ group elements in the ciphertext; in the worst case of $\ell = w = L$ this means $(n + 1)L + 2$ exponentiations and group elements. (The pairing $\hat{e}(h_1, g_2)$ can be precomputed.) Decryption requires the computation of $\ell + 1$ pairings.

In terms of efficiency, the Waters-WIBE scheme performs well enough to be considered for use in practice, but definitely leaves room for improvement. The main problem is the dependency of the scheme on n, the bit length of identity strings. In practice, one would typically use the output of a collision-resistant hash function as identity strings, so that $n = 160$ for a reasonable level of security. We note that the techniques of [CS05, Nac07] could be applied to trade a factor d in efficiency against the loss of a factor of 2^{Ld} in the tightness of the reduction.

We now prove the security of the Waters-WIBE, relative to the security of the Waters-HIBE. This proof provides a template for the proofs of the security theorems for the BB and BBG WIBE's mentioned above. We reduce the security of the Waters-WIBE to the security of the Waters-HIBE. The security of the latter scheme, as has already been mentioned, is believed to reduce to the security of the BDDH problem (see Section E.2.8).

Theorem E.5.3 If there exists a (t, q_K, ϵ)-adversary against the IND-WID-CPA security of the Waters-WIBE scheme (with hierarchy depth L) then there exists a (t', q'_K, ϵ')-adversary against the IND-HID-CPA security of the HIBE scheme, where

$$t' \leq t + Ln(1 + q_K) \cdot t_{exp},\ q'_K \leq q_K \text{ and } \epsilon' \geq \epsilon / 2^L,$$

and t_{exp} is the time it takes to perform an exponentiation in $\mathbb{G}$.

Proof: Suppose there exists a (t, q_K, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CPA security of the Waters-WIBE scheme. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the IND-HID-CPA security of the Waters-HIBE.

The intuitive idea behind the proof is that $\mathcal{B}$ guesses the levels in which the challenge pattern contains wildcards. Any query that $\mathcal{A}$ makes is passed by $\mathcal{B}$ to its own oracles after stripping out the levels corresponding to wildcards in the challenge pattern. To this end, we construction a 'projection" map $\pi : \{1, \ldots, L\} \rightarrow \{1, \ldots, L\}$. Suppose that $\bar{P}^* \in \{\varepsilon, *\}^L$ is $\mathcal{B}$'s guess for the wildcard positions in the challenge pattern. Define $\bar{P}^*_{\leq i}$ to be equal to the first i components of $\bar{P}^*$ and define π as

$$\pi(i) = \begin{cases} 0 & \text{if } i \in W(\bar{P}) \\ i - |W(\bar{P}^*_{\leq i})| & \text{if } i \notin W(\bar{P}) \end{cases}$$

$\mathcal{B}$ is an adversary against the Waters-HIBE scheme. We denote parameters associated with the HIBE scheme using tildes. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ takes as input the master public key of the HIBE scheme $\tilde{mpk} = (\tilde{g}_1, \tilde{g}_2, \tilde{h}_1, \tilde{u}_{1,0}, \ldots, \tilde{u}_{L,n})$.

2. $\mathcal{B}_1$ computes $\bar{P} = (\bar{P}_1, \ldots, \bar{P}_L) \xleftarrow{\$} \{\varepsilon, *\}^L$.

3. $\mathcal{B}_1$ computes the master public key $mpk = (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})$ as follows:

$$\begin{array}{l} g_1 \leftarrow \tilde{g}_1 \qquad g_2 \leftarrow \tilde{g}_2 \qquad h_1 \leftarrow \tilde{h}_1 \\ u_{i,j} \leftarrow \tilde{u}_{\pi(i),j} \text{ if } i \notin W(\bar{P}) \text{ and } j = 1, \ldots, n \\ u_{i,j} \leftarrow g_1^{\alpha_{i,j}} \text{ if } i \in W(\bar{P}),\ j = 1, \ldots, n \text{ and } \alpha_{i,j} \xleftarrow{\$} \mathbb{Z}_p \end{array}$$

4. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle on input $ID = (ID_1, \ldots, ID_\ell)$ then $\mathcal{B}_1$ constructs an identity $\tilde{ID} = (\tilde{ID}_1, \ldots, \tilde{ID}_{\tilde{\ell}})$ by setting $\tilde{ID}_{\pi(i)} \leftarrow ID_i$ for each $i \in W(\bar{P}^*_{\leq \ell})$. $\mathcal{B}_1$ queries its key derivation oracle on $\tilde{ID}$ and receives $(\tilde{d}_0, \ldots, \tilde{d}_{\tilde{\ell}})$. $\mathcal{B}_1$ reconstructs the decryption key $d_{ID} = (d_0, \ldots, d_\ell)$ for ID as:

$$\begin{array}{lcll} d_0 & \leftarrow & \tilde{d}_0 \prod_{i \in W(\bar{P}^*_{\leq \ell})} (u_{i,0} \prod_{j \in [ID_i]} u_{i,j})^{r_i} & \text{for } r_i \xleftarrow{\$} \mathbb{Z}_p \\ d_i & \leftarrow & d_{\pi(i)} & \text{if } i \notin W(\bar{P}^*_{\leq \ell}) \\ d_i & \leftarrow & g_1^{r_i} & \text{if } i \in W(\bar{P}^*_{\leq \ell}) \end{array}$$

   $\mathcal{B}_1$ returns the key d_{ID} to $\mathcal{A}_1$. $\mathcal{A}_1$ outputs two equal-length messages (m_0, m_1) and a challenge pattern $P^* = (P^*_1, \ldots, P^*_{\ell^*})$.

5. If $\bar{P}^*_{\leq \ell^*}$ and P^* do not have wildcards in exactly the same positions, then $\mathcal{B}_1$ aborts. Otherwise, $\mathcal{B}_1$ computes a challenge identity $\tilde{ID}^* = (\tilde{ID}^*_1, \ldots, \tilde{ID}^*_{\tilde{\ell}^*})$ by setting $\tilde{ID}^*_i \leftarrow P^*_i$ for all $i \notin W(P^*)$. $\mathcal{B}_1$ outputs the challenge identity $\tilde{ID}^*$ and the two messages (m_0, m_1).

The challenger will now encrypt m_β under the identity $\tilde{ID}^*$ using the Waters-HIBE (for $\beta \xleftarrow{\$} \{0, 1\}$). This results in a ciphertext $\tilde{C}^* = (\tilde{C}^*_1, \tilde{C}^*_{2,1}, \ldots, \tilde{C}^*_{2,\tilde{\ell}^*}, \tilde{C}^*_3)$ which is input to the algorithm $\mathcal{B}_2$ described below:

1. $\mathcal{B}_2$ computes a challenge WIBE ciphertext $C^* = (P^*, C_1^*, C_{2,1}^*, \ldots, C_{2,L}^*, C_3^*)$ as follows:

$$\begin{array}{lcll} C_1^* & \leftarrow & \tilde{C}_1^* & \\ C_{2,i}^* & \leftarrow & \tilde{C}_{2,\pi(i)}^* & \text{if } i \notin W(P^*) \\ C_{2,i}^* & \leftarrow & (C_1^{*\alpha_{i,0}}, \ldots, C_1^{*\alpha_{i,n}}) & \text{for } i \in W(P^*) \\ C_3^* & \leftarrow & \tilde{C}_3^* & \end{array}$$

2. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes a key derivation oracle query, then $\mathcal{B}_2$ answer its queries as before. $\mathcal{A}_2$ outputs a guess β'.

3. $\mathcal{B}_2$ outputs β'.

We make several observations about the adversary $\mathcal{B}$. First, note that $\mathcal{B}$ cannot correctly guess the bit β' unless it correctly guesses the locations of the wildcards in the challenge pattern. This happens with probability at least $1/2^L$. Second, we observe that if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then $\mathcal{B}$ correctly simulates the key derivation oracle and challenge ciphertext for $\mathcal{A}$. Furthermore, if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then any legal key derivation oracle query that $\mathcal{A}$ makes results in a legal key derivation oracle query made by $\mathcal{B}$. This is because for any identity $ID \not\in_* P^*$ there must exist an index i such that $P_i^* \neq *$ and $ID_i \neq P_i^*$. Hence, the "projected" identity $\tilde{ID}$ has $\tilde{ID}_{\pi(i)} = ID_i \neq P_i^* = \tilde{ID}_{\pi(i)}^*$. Hence, if $\mathcal{B}$ correctly guesses the position of the wildcards in the challenge ciphertext, then $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. This leads to the results of the theorem. ∎

Note that the proof above loses a factor of 2^L in the security reduction. This limits the secure use of the scheme in practice to very small (logarithmic) hierarchy depths, but this was already the case for the Waters-HIBE scheme, which loses a factor $(nq_K)^L$ in its reduction to the BDDH problem. Alternatively, if we only consider patterns with a single sequence of consecutive wildcards, for example $(ID_1, *, *, *, ID_5)$ or $(ID_1, *, *)$, then we only lose a factor of L^2 when reducing to the Waters-HIBE scheme. If we consider the selective-identity notion, there is no need to guess the challenge pattern, so we do not lose any tightness with respect to the Waters-HIBE scheme. In addition, the Waters-HIBE scheme would itself also have a tight security reduction to the BDDH problem in the selective-identity notion.

E.5.4 Converting Selective-Identity Security to Full Security

As observed by Boneh and Boyen [BB04a] for the case of IBE schemes and by Boneh, Boyen, and Goh [BBG05] for the case of HIBE schemes, any HIBE scheme that is selective-identity (IND-sHID) secure can be transformed into a HIBE scheme that is fully (IND-HID) secure in the random oracle model. The transformation only works for small hierarchy depths though, since the proof loses a factor $O(q_H^L)$ in reduction tightness. We show here that the same transformation works for the case of WIBE schemes at a similar cost of a factor $O(q_H^L)$ in reduction.

Let $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a WIBE scheme with maximum hierarchy depth L. We construct a WIBE scheme $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ where KeyDer', $\mathsf{Encrypt}'$, and $\mathsf{Decrypt}'$ are identical to KeyDer, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ with the exception that the identity/pattern is input to a hash function before it is input to the relevant algorithm. A pattern $P = (P_1, \ldots, P_\ell)$ is transformed into a pattern $P' = (P_1', \ldots, P_\ell')$ where

$$P_i' \leftarrow \begin{cases} H_i(P_i) & \text{if } P_i \neq * \\ * & \text{otherwise,} \end{cases}$$

where $H_i : \{0,1\}^* \to \mathcal{ID}$ (for $1 \leq i \leq L$) are independent hash functions (modelled as distinct random oracles) and $\mathcal{ID}$ is an appropriately sized subset of the allowable identities for the original WIBE scheme.[1]

Theorem E.5.4 In the random oracle model, suppose that there exists a (t, q_K, q_H, ϵ)-adversary against the IND-WID-CPA security of Π' (with hierarchy depth L) then there exists a (t', q_K, ϵ')-adversary against the IND-sWID-CPA security of Π, where $t' \leq t$ and

$$\epsilon' \geq \frac{\epsilon}{(L+1)(q_H + q_K L + 1)^L} - \frac{(q_H + q_K L + 1)^2}{|\mathcal{ID}|} .$$

Proof: Suppose there exists a (t, q_K, q_H, ϵ)-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CPA security of Π'. We construct an IND-sWID-CPA adversary $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ against Π that uses $\mathcal{A}$ as a subroutine. The algorithm $\mathcal{B}_0$ runs as follows:

1. $\mathcal{B}_0$ chooses $\hat{\ell}^* \xleftarrow{\$} \{0, 1, \ldots, L\}$ and $\hat{ctr} \xleftarrow{\$} \{0, 1, \ldots, q_H + q_K L + 1\}$. $\mathcal{B}_0$ computes the challenge pattern $\hat{P}^* \leftarrow (\hat{P}^*_1, \ldots, \hat{P}^*_{\hat{\ell}^*})$ where

$$\hat{P}^*_i \leftarrow \begin{cases} * & \text{if } \hat{ctr} = 0 \\ ID & \text{if } \hat{ctr} \neq 0 \text{ where } ID \xleftarrow{\$} \mathcal{ID} \end{cases}$$

   $\mathcal{B}_0$ outputs $\hat{P}^*$.

The challenger now issues the master public key mpk to the adversary. Algorithm $\mathcal{B}_1$ run as follows:

1. $\mathcal{B}_1$ receives the master public key mpk.
2. $\mathcal{B}_1$ initialises a set of lists T_i to answer the random oracle queries for the hash function H_i. These lists are initially empty. For each list, $\mathcal{B}_1$ initialises a counter $ctr_i \leftarrow 1$.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. $\mathcal{B}_1$ answers $\mathcal{A}_1$'s oracle queries as follows:
   - Suppose $\mathcal{A}_1$ queries the random oracle H_i on input ID. If $T_i[ID]$ is defined, then $\mathcal{B}_1$ returns $T_i[ID]$. Otherwise, if $ctr_i = \hat{ctr}_i$, then $\mathcal{B}_1$ sets $T_i[ID] \leftarrow \hat{P}^*_i$, else $\mathcal{B}_1$ sets $T_i[ID] \xleftarrow{\$} \mathcal{ID}$. In either case, $\mathcal{B}_1$ increments ctr_i by one and returns $T_i[ID]$.
   - Suppose $\mathcal{A}_1$ queries the key derivation oracle on $ID = (ID_1, \ldots, ID_\ell)$. $\mathcal{B}_1$ computes the hashed identity $ID' = (ID'_1, \ldots, ID'_\ell)$ where $ID'_i \leftarrow H_i(ID_i)$ using the random oracle algorithm defined above. $\mathcal{B}_1$ queries its own key derivation oracle on the input ID' and returns to the result to $\mathcal{A}_1$.

   $\mathcal{A}_1$ terminates by outputting a challenge pattern $P^* = (P^*_1, \ldots, P^*_{\ell^*})$ and two equal-length messages (m_0, m_1).
4. If $\ell^* \neq \hat{\ell}^*$, if there exists $i \in W(\hat{P}^*)$ such that $P^*_i \neq *$, or if there exists $1 \leq i \leq \ell^*$ such that $i \notin W(\hat{P}^*)$ and $H_i(P^*_i) \neq \hat{P}^*_i$, then $\mathcal{B}_1$ aborts.
5. $\mathcal{B}_1$ outputs the two messages (m_0, m_1).

[1] These L independent random oracles $(H_1, \ldots, H_L)$ can easily be constructed from a single random oracle H, e.g. by setting $H_i(\cdot) = H([i] \| \cdot)$ where $[i]$ is a fixed-length representation of the integer i.

The challenger computes the challenge ciphertext C^* (which is the encryption of m_β for some randomly chosen $\beta \xleftarrow{\$} \{0,1\}$). This value is input to algorithm $\mathcal{B}_2$ which runs as follows:

1. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input C^*. If $\mathcal{A}_2$ makes any oracle query, then they are answered as above. $\mathcal{A}_2$ outputs a bit β'.
2. $\mathcal{B}_2$ outputs β'.

$\mathcal{B}$ wins the IND-sWID-CPA game if (1) $\mathcal{A}$ wins the IND-WID-CPA game; (2) $\mathcal{B}$ does not abort because the challenge pattern it outputs is incorrect; (3) $\mathcal{A}$ does not force $\mathcal{B}$ to make an illegal key derivation oracle query. The idea is that the counters $\hat{ctr}_i$ are $\mathcal{B}$'s guess as to which oracle query will define the challenge patterns (where a counter values of $\hat{ctr}_i = 0$ means that that position is a wildcard). We require that for each of the hash oracles provides no collisions – i.e. for each $ID \neq ID'$ we have $H_i(ID) \neq H_i(ID')$. Since such a collision could only occur by accident, the probability is bounded by $(q_H + q_K L + 1)^2/|\mathcal{ID}|$ as there exists at most $q_H + q_K L + 1$ entries in all the lists. We exclude the possibility this occurs by losing an additive factor of $(q_H + q_K L)^2/|\mathcal{ID}|$ in the security reduction.

Furthermore, we require that the algorithm $\mathcal{B}$ correctly identifies the pattern that $\mathcal{A}$ outputs. Since the values are chosen at random, we have that $\hat{\ell}^* = \ell^*$ with probability $1/(L+1)$ and that the $\hat{ctr}_i$ value will be correct with probability $1/(q_H + q_K L + 1)$. If $\mathcal{B}$ correctly guesses these values and there are no hash collisions, then $\mathcal{A}$ will never force $\mathcal{B}$ to make an illegal key derivation query. Hence, the result of the theorem holds. ∎

The above theorem is easily seen to extend to the case of converting an IND-sWID-CCA scheme into an IND-WID-CCA scheme, with an appropriate alteration of the error term in the advantage statement; to take into account the number of decryption oracle queries. Indeed adversary $\mathcal{B}$ is modified so that when it obtains a decryption query it first hashes the identities to produce a decryption query suitable for $\mathcal{A}$. Such a simulation will fail if and only if the hashed identity is equivalent to the challenge identity for $\mathcal{A}$, but this would imply a collision in the random oracle.

E.6 IND-WID-CCA Secure WIBEs

In this section, we present constructions for IND-WID-CCA secure WIBEs. We present one generic transform from an IND-WID-CPA WIBE into an IND-WID-CCA WIBE based on the Canetti-Halevi-Katz transform [CHK04] and a generic random-oracle-based transform from an OW-WID-CPA WIBE into an IND-WID-CCA WIB-KEM based on a transform of Dent [Den03].

E.6.1 The Canetti-Halevi-Katz Transform

In this section, we construct a variant of the Canetti-Halevi-Katz transform [CHK04] to convert an IND-WID-CPA secure WIBE with hierarchy depth $L+1$ into an IND-WID-CCA secure WIBE with hierarchy depth L, using a one-time signature scheme (see Section E.2.3).

In order to complete this transform, we will make liberal use of an "encoding" function Encode. We will need to restrict the space of allowable identities. We assume that "$-$" represents some fixed, public-known allowable identity for the CPA scheme; we will deliberately exclude "$-$" from the space of allowable identities in the CCA scheme. We assume that 1^k is an allowable identity in the CCA scheme. We then encode a pattern $P = (P_1, \ldots, P_\ell)$ and a verification key vk as the $L+1$ level identity:

$$\mathsf{Encode}(P, vk) = (P_1, \ldots, P_\ell, -, \ldots, -, vk)\,.$$

We define a similar map for identities (interpreted as patterns without wildcards).

Given an IND-WID-CPA WIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with hierarchy depth $L+1$, we define an IND-WID-CCA WIBE $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}', \mathsf{Decrypt}')$ with hierarchy depth L. This scheme is described in Figure E.7. The encryption algorithm now produces ciphertexts which are (a) encrypted under the pattern $\mathsf{Encode}(P, vk)$ for a randomly generated $(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$, and (b) signed using sk. The decryption algorithm checks the signature and (if correct) decrypts the ciphertext using a key for an identity which matches $\mathsf{Encode}(P, vk)$ (using the valid identity 1^k in place of wildcards).

Algorithm $\mathsf{Encrypt}'(mpk, P, m)$:
  $(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$
  $P' \leftarrow \mathsf{Encode}(P, vk)$
  $C' \xleftarrow{\$} \mathsf{Encrypt}(mpk, P', m)$
  $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, (P, C'))$
  $C \leftarrow (vk, C', \sigma)$
  Return C

Algorithm $\mathsf{Decrypt}'(d_{ID}, C)$:
  Parse C as (vk, C', σ)
  If $\mathsf{Verify}(vk, C', \sigma) = \perp$ then return $\perp$
  For i equals 1 to $|P| - |ID|$
    If $P_{|ID|+i} \neq *$ then $ID'_i \leftarrow P_{|ID|+i}$
    If $P_{|ID|+i} = *$ then $ID'_i \leftarrow 1^k$
  For i equals 1 to $L - |P|$
    $ID'_{|P|-|ID|+i} \leftarrow -$
  $ID'_{L-|ID|+1} \leftarrow vk$
  $d \xleftarrow{\$} \mathsf{KeyDer}(d_{ID}, ID')$
  $m \leftarrow \mathsf{Decrypt}(d, C)$
  Return m

Figure E.7: The Canetti-Halevi-Katz transform.

Theorem E.6.1 Suppose that there exists a (t, q_K, q_D, ϵ)-adversary against the IND-WID-CCA security of the WIBE Π' then there exists a $(t_w, q_K + q_D, \epsilon_w)$-adversary against the IND-WID-CPA security of Π and a (t_s, ϵ_s)-adversary against the one-time unforgeability of the signature scheme, where

$$\begin{aligned} t_w &\leq t + t_{\mathsf{SigGen}} + t_{\mathsf{Sign}} + q_D(t_{\mathsf{Verify}} + t_{\mathsf{Decrypt}}) , \\ t_s &\leq t + t_{\mathsf{Setup}} + t_{\mathsf{Encrypt}} + q_K \cdot t_{\mathsf{KeyDer}} + q_D \cdot t_{\mathsf{Decrypt}} , \\ \epsilon &\geq \epsilon_w + 2\epsilon_s , \end{aligned}$$

where $t_{\mathtt{ALG}}$ is the time to execute the algorithm $\mathtt{ALG}$.

Proof: The proof closely follows that of [CHK04]. Let $\mathcal{A}$ be an IND-WID-CCA adversary against the scheme Π'. Suppose P^* is the challenge pattern that $\mathcal{A}$ chooses and (vk^*, C^*, σ^*) is the challenge ciphertext that $\mathcal{A}$ receives during an execution of the attack game. Let FORGE be the event that at some point during its execution $\mathcal{A}$ queries the decryption oracle on an identity $ID \in_* P^*$ and a ciphertext of the form (vk^*, C, σ) such that the algorithm $\mathsf{Verify}(vk^*, C, \sigma)$ returns $\top$. Then we have that $\mathcal{A}$'s advantage is

$$\left|2 \cdot \Pr[\mathcal{A} \text{ wins}] - 1/2\right| \leq \left|2 \cdot \Pr[\mathcal{A} \text{ wins} \mid \neg\textsc{Forge}] - 1\right| + 2 \cdot \Pr[\textsc{Forge}] .$$

Claim E.6.2 $\Pr[\textsc{Forge}] \leq \epsilon_s$.

We describe an algorithm $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ which breaks the one-time unforgeability of the signature scheme if the event FORGE occurs. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ receives vk^* as input.
2. $\mathcal{B}_1$ generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a decryption or key derivation oracle query, then $\mathcal{B}_1$ answers it using its knowledge of the master private key msk. $\mathcal{B}_1$ outputs a challenge pattern P^* and two equal-length messages (m_0, m_1).
4. If $\mathcal{A}_1$ submitted a decryption oracle query (vk^*, C, σ) for which $\mathsf{Verify}(vk^*, C, \sigma) = \top$, then $\mathcal{B}_1$ chooses a ciphertext $C^* \neq C$ and returns C^*. This is known as the error event.
5. Otherwise, $\mathcal{B}_1$ chooses $\beta \xleftarrow{\$} \{0,1\}$, computes $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathsf{Encode}(P^*, vk^*), m_\beta)$ and returns C^*.

The challenger then computes a signature σ^* on the "message" C^*. This is input to the algorithm $\mathcal{B}_2$ described as follows:

1. $\mathcal{B}_2$ receives σ^* as input.
2. If the error event occurred during the first phase, then $\mathcal{B}_2$ outputs (C, σ).
3. Otherwise $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input (vk^*, C^*, σ^*). If $\mathcal{A}_2$ makes a key derivation or decryption oracle query, then $\mathcal{B}_2$ answers them using its knowledge of the master private key msk. $\mathcal{B}_2$ outputs a bit β'.
4. If $\mathcal{A}_2$ submitted a decryption oracle query (vk^*, C, σ) for which $\mathsf{Verify}(vk^*, C, \sigma) = \top$, then $\mathcal{B}_2$ outputs (C, σ). Otherwise $\mathcal{B}_2$ outputs the error symbol $\perp$.

Algorithm $\mathcal{B}$ is designed to output a valid forgery if the event FORGE occurs. If $\mathcal{A}_1$ makes a valid decryption oracle query on (vk^*, C, σ), then the error event occurs, and $\mathcal{B}$ trivially wins. If $\mathcal{A}_2$ makes a valid decryption oracle query on (vk^*, C, σ), then, since $\mathcal{A}_2$ is forbidden from making a decryption oracle query on (vk^*, C^*, σ^*), $\mathcal{B}$ wins after $\mathcal{A}$ finishes its execution. Hence, we have $\epsilon_s = \Pr[\,\text{FORGE}\,]$.

Claim E.6.3 $|2 \cdot \Pr[\,\mathcal{A} \text{ wins} \mid \neg\text{FORGE}\,] - 1| \leq \epsilon_w$.

We describe an algorithm $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ which breaks the IND-WID-CPA security of the WIBE scheme Π whenever $\mathcal{A}$ wins and FORGE did not occur. Algorithm $\mathcal{B}'_1$ runs as follows:

1. $\mathcal{B}'_1$ receives a master public key mpk as input.
2. $\mathcal{B}'_1$ generates $(vk^*, sk^*) \xleftarrow{\$} \mathsf{SigGen}$.
3. $\mathcal{B}'_1$ run $\mathcal{A}_1$ on mpk. If $\mathcal{A}_1$ makes a key derivation oracle query on identity ID, then $\mathcal{B}_1$ makes a key derivation oracle query on ID and returns the result. If $\mathcal{A}_1$ makes a decryption oracle query on identity ID and ciphertext (vk, C, σ), then $\mathcal{B}'_1$ returns $\perp$ if $vk = vk^*$ or if $\mathsf{Verify}(vk, C, \sigma) = \perp$. Otherwise, $\mathcal{B}'_1$ computes the extension identity ID' required so that $ID \| ID'$ matches the pattern $\mathsf{Encode}(P, vk)$ as in the decryption algorithm, queries the key extraction algorithm on $ID \| ID'$ to obtain a decryption key d and returns $\mathsf{Decrypt}(d, C)$. $\mathcal{A}_1$ outputs a pattern P^* and two equal-length messages (m_0, m_1).
4. $\mathcal{B}'_1$ returns the challenge pattern $\mathsf{Encode}(P^*, vk^*)$ and the messages (m_0, m_1).

The challenger will pick a random $\beta \xleftarrow{\$} \{0,1\}$ and computes the ciphertext

$$C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, \mathsf{Encode}(P^*, vk^*), m_\beta).$$

This ciphertext is input to the algorithm $\mathcal{B}_2'$ below:

1. $\mathcal{B}_2'$ receives the ciphertext C. $\mathcal{B}_2'$ computes $\sigma^* \xleftarrow{\$} \mathsf{Sign}(sk^*, C^*)$.
2. $\mathcal{B}_2'$ runs $\mathcal{A}_2$ on the ciphertext (vk^*, C^*, σ^*). All oracle queries are answered in exactly the same way as in the the first phase. $\mathcal{A}_2$ outputs a bit β'.
3. $\mathcal{B}_2'$ outputs β'.

It is clear that as long as $\mathcal{B}'$ does not make an illegal key derivation oracle query, then $\mathcal{B}'$ wins if and only if $\mathcal{A}$ wins (assuming that FORGE does not occur). $\mathcal{B}'$ may make key derivation oracle queries in response to $\mathcal{A}$ making a key derivation oracle query or a decryption oracle query. If $\mathcal{A}$ makes a decryption oracle query on an identity ID and ciphertext (vk, C, σ) then $\mathcal{B}'$ makes a key derivation query on $\mathsf{Encode}(ID \| ID', vk)$; however $\mathsf{Encode}(ID \| ID', vk) \not\in_* \mathsf{Encode}(P^*, vk^*)$ as both encodings are $(L+1)$-bits long and $vk \neq vk^*$ since FORGE does not occur. Furthermore, if $\mathcal{A}$ makes a key derivation oracle query on an identity ID then, by definition, we have $ID \not\in_* P^*$. We need to show that $ID \not\in_* \mathsf{Encode}(P^*, vk^*)$. This is true as:

- if $|ID| > |P^*|$ then ID and $\mathsf{Encode}(P^*, vk^*)$ do not agree at level $|P^*|+1$ (where $\mathsf{Encode}(P^*, vk^*)$ is defined to be "$-$" and ID cannot be defined to be "$-$" since it was excluded from the message space);
- if $|ID| \leq |P^*|$ then $ID \not\in_* \mathsf{Encode}(P^*, vk^*)$ as $\mathsf{Encode}(P^*, vk^*)_i = P_i^*$ for levels $1 \leq i \leq |ID|$ and $ID \not\in_* P^*$.

Hence, $\mathcal{A}$ never forces $\mathcal{B}'$ to make an illegal key derivation oracle query and so $\mathcal{B}'$ wins whenever $\mathcal{A}$. Thus,

$$\left|2 \cdot \Pr\left[\mathcal{A} \text{ wins} \mid \neg \text{FORGE}\right] - 1\right| \leq \epsilon_w\,.$$

A combination of the two claims gives the theorem. ∎

Applying the transformation to Waters-WIBE.

We may optimise the CHK transform in the particular case of the Wates-WIBE scheme describe in Section E.5.3. In particular, there is no implicit functional reason why we have to fix the encoded identity using "$-$" strings, as it is possible to determine a key for which the $(L+1)$-th level is fixed to vk while leaving lower levels undetermined. In particular, we obtain the scheme given in Figure E.8 which is IND-CCA secure and has depth L. We assume (for simplicity) that verification keys vk are n-bits long.

E.6.2 The Dent KEM Transform

One approaching to building systems secure against adaptive chosen ciphertext attacks is to transform a weakly-secure (OW-WID-CPA) WIBE scheme into a strongly-secure (IND-WID-CCA) WIB-KEM scheme. This obviously gives rise to an IND-WID-CCA WIBE scheme when combined with a suitably secure DEM (see Sections E.2.9 and E.4). We apply an analogue of the transformation of Dent [Den03].

Algorithm Setup:
$g_1, g_2 \xleftarrow{\$} \mathbb{G}$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$
$h_1 \leftarrow g_1^{\alpha}$; $h_2 \leftarrow g_2^{\alpha}$
$u_{i,j} \xleftarrow{\$} \mathbb{G}$ for $i = 1, \ldots, L+1; j = 0 \ldots n$
$mpk \leftarrow (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L+1,n})$
$msk \leftarrow h_2$
Return (mpk, msk)

Algorithm $\mathsf{KeyDer}(d_{(ID_1,\ldots,ID_\ell)}, ID_{\ell+1})$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
$r_{\ell+1} \xleftarrow{\$} \mathbb{Z}_p$
$d'_0 \leftarrow d_0 \cdot F_{\ell+1}(ID_{\ell+1})^{r_{\ell+1}}$
$d'_{\ell+1} \leftarrow g_1^{r_{\ell+1}}$
Return $(d'_0, d_1, \ldots, d_\ell, d'_{\ell+1})$

Algorithm $\mathsf{Encrypt}(mpk, P, m)$:
Parse P as $(P_1, \ldots, P_\ell)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $C_1 \leftarrow g_1^r$
For $i = 1 \ldots \ell$ do
  If $i \notin W(P)$ then $C_{2,i} \leftarrow F_i(ID_i)^r$
  If $i \in W(P)$ then $C_{2,i} \leftarrow (u_{i,0}^r, \ldots, u_{i,n}^r)$
$(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$
$C_{2,L+1} \leftarrow F_{L+1}(vk)^r$
$C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^r$
$\sigma \leftarrow \mathsf{Sign}(sk, (P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3))$
Return $(vk, P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3, \sigma)$

Algorithm $\mathsf{Decrypt}(d_{(ID_1,\ldots,ID_\ell)}, C)$:
Parse $d_{(ID_1,\ldots,ID_\ell)}$ as $(d_0, \ldots, d_\ell)$
Parse C as $(vk, P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3, \sigma)$
If $\mathsf{Verify}(vk, (P, C_1, C_{2,1}, \ldots, C_{2,\ell}, C_{2,L+1}, C_3), \sigma) = \perp$ then
  Return $\perp$
For $i = 1, \ldots, \ell$ do
  If $i \notin W(P)$ then $C'_{2,i} \leftarrow C_{2,i}$
  If $i \in W(P)$ then
    Parse $C_{2,i}$ as $(v_0, \ldots, v_n)$
    $C'_{2,i} \leftarrow v_0 \prod_{i \in [ID_i]} v_i$
$m' \leftarrow C_3 \cdot \frac{\hat{e}(g_1, C_{2,L+1}) \cdot \prod_{i=1}^{\ell} \hat{e}(d_i, C'_{2,i})}{\hat{e}(C_1, F_{L+1}(vk)) \cdot \hat{e}(C_1, d_0)}$
Return m'

Figure E.8: The IND-WID-CCA Waters WIBE scheme.

Suppose $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an OW-WID-CPA WIBE scheme (see Section E.3.2) with a finite message space $\mathcal{M}$. We assume that the $\mathsf{Encrypt}$ algorithm uses random values taken from a set $\mathcal{R}$. We can write $\mathsf{Encrypt}$ as a deterministic algorithm $C \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$ where $r \xleftarrow{\$} \mathcal{R}$. We require that the scheme satisfies a notion of randomness called γ-uniformity.

Definition E.6.4 A WIBE scheme Π is γ-uniform if for all master public keys mpk that could be output by the key generation algorithm, for all patterns P, for all messages m and ciphertexts C, we have

$$\Pr\left[\,\mathsf{Encrypt}(mpk, P, m; r) = C\,\right] \leq \gamma,$$

where the probability is taken over the choice of the randomness r used in the encryption function.

The only difficulty in applying the method of Dent [Den03] is that we must re-encrypt the recovered message as an integrity check. In the WIBE setting, this means we must know the pattern under which the message was originally encrypted. Recall, that the set $W(C) = \{i \in \mathbb{Z} : P_i = *\}$ of the pattern P used to encrypt the message, along with the length ℓ of the pattern, is easily derived from the ciphertext. We use this information to give an algorithm P, which on input (ID, C), where C is a ciphertext and $ID = (ID_1, \ldots, ID_\ell)$, returns the pattern $P = (P_1, \ldots, P_\ell)$ where

$$P_i = \begin{cases} * & \text{if } i \in W(C) \\ id_i & \text{if } i \notin W(C) \end{cases}$$

We transform the WIBE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with a finite message space $\mathcal{M}$ and hierarchy depth L into a WIB-KEM scheme $\Pi' = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Encap}, \mathsf{Decap})$ using two hash functions:

$$H_1 : (\{0,1\}^n \cap \{*\})^* \times \{0,1\}^* \to \mathcal{R}$$

and

$$H_2 : \{0,1\}^* \rightarrow \{0,1\}^\lambda .$$

The complete scheme is given in Figure E.9.

Algorithm $\mathsf{Encap}(mpk, P)$:
$m \leftarrow \mathcal{M}$
$r \leftarrow H_1(P, m)$
$C \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$
$K \leftarrow H_2(m)$
Return (C, K)

Algorithm $\mathsf{Decap}(d_{ID}, C)$:
$m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$
$P \leftarrow \mathsf{P}(ID, C)$
$r \leftarrow H_1(P, m)$
$C' \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$
If $C = C'$ then return m
Otherwise return $\perp$

Figure E.9: The Dent transform.

Theorem E.6.5 Suppose that there exists a $(t, q_K, q_D, q_H, \epsilon)$-adversary, in the random oracle model, against the IND-WID-CCA security of the WIB-KEM Π' then there exists a (t', q_K, ϵ')-adversary against the OW-WID-CPA security of the WIBE Π, where

$$\begin{aligned} \epsilon' &\geq \frac{\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)}{q_H + q_D} \\ t' &\leq t + q_H t_{\mathsf{Encrypt}} \end{aligned}$$

where t_{Encrypt} is the time taken to perform an encryption, Π has finite message space $\mathcal{M}$, and Π is γ-uniform.

Proof: Suppose there exists a $(t, q_K, q_D, q_H, \epsilon)$-adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the IND-WID-CCA security of the WIB-KEM in the random oracle model. We construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the OW-WID-CPA security of the WIBE. The algorithm $\mathcal{B}_1$ runs as follows:

1. $\mathcal{B}_1$ receives a master public key mpk.
2. $\mathcal{B}_1$ initialises three lists T_1, E_1, and T_2 which are initially set to be empty.
3. $\mathcal{B}_1$ runs $\mathcal{A}_1$ on mpk. $\mathcal{B}_1$ answers $\mathcal{A}_1$'s oracle queries as follows:
   - Suppose $\mathcal{A}_1$ queries the H_1-oracle on input (P, m). If $T_1[P, m]$ is defined, $\mathcal{B}_1$ returns $T_1[P, m]$. Otherwise, $\mathcal{B}_1$ chooses $r \xleftarrow{\$} \mathcal{R}$, sets $T_1[P, m] \leftarrow r$, sets $E_1[P, m] \leftarrow \mathsf{Encrypt}(mpk, P, m; r)$, and returns r.
   - Suppose $\mathcal{A}_1$ queries the H_2-oracle on input r. If $T_2[r]$ is defined, $\mathcal{B}_1$ returns $T_2[r]$. Otherwise, $\mathcal{B}_1$ chooses $K \xleftarrow{\$} \{0,1\}^\lambda$, sets $T_2[r] \leftarrow K$, and returns K.
   - Suppose $\mathcal{A}_1$ queries the key derivation oracle on the input ID. $\mathcal{B}_1$ forwards this request to its own key derivation oracle and returns the result.
   - Suppose $\mathcal{A}_1$ queries the decryption oracle on the identity ID and the ciphertext C. $\mathcal{B}_1$ searches the list T_1 for an entry $C = E_1[P, m]$ where $P = \mathsf{P}(ID, C)$. If no such entry exists, then $\mathcal{B}_1$ returns $\perp$. Otherwise, $\mathcal{B}_1$ computes $K \leftarrow H_2(m)$ as above and returns K.

   The adversary outputs a challenge pattern P^*.
4. $\mathcal{B}_1$ outputs the challenge pattern P^*.

The challenger then computes a challenge encryption $C^* \xleftarrow{\$} \mathsf{Encrypt}(mpk, P^*, m^*; r^*)$ for $m^* \xleftarrow{\$} \mathcal{M}$ and $r^* \xleftarrow{\$} \mathcal{R}$. This ciphertext is input to the algorithm $\mathcal{B}_2$:

1. $\mathcal{B}_2$ receives C^*.
2. $\mathcal{B}_2$ generates $K^* \xleftarrow{\$} \{0,1\}^\lambda$.
3. $\mathcal{B}_2$ runs $\mathcal{A}_2$ on the input (C^*, K^*). If $\mathcal{A}_2$ queries any oracle, then $\mathcal{B}_2$ answers these queries as before. $\mathcal{A}_2$ outputs a bit β'.
4. $\mathcal{B}_2$ randomly chooses a defined entry for one of the hash functions, either $T_1[P, m]$ or $T_2[m]$, and outputs m.

The basic strategy of this security proof is to take advantage of the fact that the only way that $\mathcal{A}$ can determine if C^* is an encapsulation of K^* is to query the H_2-oracle on m^*. However, we first have to show that the simulated hash function, key derivation, and decryption oracles are consistent with the real IND-WID-CCA game.

The simulated key derivation oracle is perfect, as is the hash function oracle, with the exception that the hash function oracle fails to respond to correctly to an H_1-oracle query on (P^*, m^*) or a H_2-oracle query on m^*. However, the decryption oracle is more problematic. There are two types of error event that can occur with the decryption oracle:

- The decryption oracle will respond incorrectly if $\mathcal{A}_1$ queries the oracle on an identity $ID \in_* P^*$ and the ciphertext C^*. However, since m^* is information theoretically hidden from $\mathcal{A}_1$, this occurs with probability at most $1/|\mathcal{M}|$.
- The decryption oracle will respond incorrectly if $\mathcal{A}$ queries the decryption oracle on an identity ID and a ciphertext C for which $T_1[P, m]$ is undefined, where $P \leftarrow \mathsf{P}(ID, C)$ and $m \leftarrow \mathsf{Decrypt}(d_{ID}, C)$, but for which

  $$C = \mathsf{Encrypt}(mpk, \mathsf{P}(ID, C), m; T_1[P, m])$$

  where $T_1[P, m]$ is randomly chosen at the end of the game if it is not defined later by an adversarial query. Since $T_1[P, m]$ is randomly chosen and Π is γ-uniform, we have that this occurs with probability γ.

We have that the probability that either of these events occurs is therefore bounded by $q_D(|\mathcal{M}|^{-1} + \gamma)$. Assuming none of these events occur, we have that the simulation is perfect unless $\mathcal{A}_1$ makes a query which defines the hash function values $T_1[P^*, m^*]$ or $T_2[m^*]$. Since $\mathcal{A}$ cannot determine whether K^* is the correct key for C^* without querying the H_2-oracle on m^*, we have that this event will occur with probability at least $\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)$. However, if this event occurs, then $\mathcal{B}$ will win the OW-WID-CPA with probability at least $1/(q_H + q_D)$ (as there exists at most $q_H + q_D$ entries on T_1 and T_2). Hence, $\mathcal{B}$ wins with probability at least

$$\epsilon' \geq \frac{\epsilon - q_D(|\mathcal{M}|^{-1} + \gamma)}{q_H + q_D}$$

which gives the theorem. ∎

Acknowledgements

We would like to thank Brent Waters, the anonymous referees of ICALP 2006, and the anonymous referees of the Journal of Cryptology for their valuable input. We also thank Mihir Bellare for pointing out the relation between WIBE and fuzzy identity-based encryption.

The first author was supported in part by the French ANR-07-TCOM-013 PACE Project. The sixth author was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government. The eighth author is supported by a Royal Society Wolfson Merit Award. The work in this paper was conducted whilst the third and sixth authors were at École normale supérieure, the fifth author was at the University of Bristol, the sixth author was a Postdoctoral Fellow of the Research Foundation – Flanders (FWO-Vlaanderen), and the second and seventh authors were at Royal Holloway College, University of London.

All authors would like to thank the support of the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT and the ICT Programme under Contract ICT-2007-216646 ECRYPT II. The information in this document reflects only the authors' views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Appendix F

Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions

Journal of Cryptology, 21(3):350–391

[ABC+08] with M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi

Abstract : *We identify and fill some gaps with regard to consistency (the extent to which false positives are produced) for public-key encryption with keyword search (PEKS). We define computational and statistical relaxations of the existing notion of perfect consistency, show that the scheme of [BDOP04] is computationally consistent, and provide a new scheme that is statistically consistent. We also provide a transform of an anonymous identity-based encryption (IBE) scheme to a secure PEKS scheme that, unlike the previous one, guarantees consistency. Finally, we suggest three extensions of the basic notions considered here, namely anonymous hierarchical identity-based encryption, public-key encryption with temporary keyword search, and identity-based encryption with keyword search.*

F.1 Introduction

There has recently been interest in various forms of "searchable encryption" [SWP00, BDOP04, Goh03, GSW04, WBDS04]. In this paper, we further explore one of the variants of this goal, namely public-key encryption with keyword search (PEKS) as introduced by Boneh, Di Crescenzo, Ostrovsky and Persiano [BDOP04].

The killer application envisaged by Boneh et al. is that of intelligent email routing. We consider emails as consisting of some header information, a body, and a list of keywords. Imagine Alice uses different electronic devices to read her email, including a pager, a PDA, and a desktop computer. Alice may prefer emails to be routed to her devices depending on the associated keywords. For example, she may like to receive emails with the keyword "urgent" on her pager, emails with the keyword "agenda" on her PDA, and all other emails on her desktop computer.

Existing mail server software could be updated to provide this type of service for plain, unencrypted email. When Bob sends an email to Alice encrypted under her public key, however,

routing becomes much harder. One option would be for Bob to leave the list of keywords unencrypted; if Bob is a colleague of Alice however, he may not like the gateway to know that he is exchanging emails with her with the keyword "personal". Alice is probably not willing to hand her decryption key to the gateway either. Rather, she would like to give the gateway some piece of trapdoor information that allows it to test whether the keyword "urgent" is among those in the list, without revealing any other information about the email to the gateway. This is exactly the type of functionality provided by a PEKS scheme. Bob can then use a standard public-key encryption scheme to encrypt the body of the email, and a PEKS scheme to separately encrypt each of the keywords.

The routing configuration of the email gateway need not be static. Alternatively, Alice could send the trapdoors for the keywords that she wants to receive at the time of login. This could be useful for checking email over a low-bandwidth connection: when Alice is at a conference, for example, she may want to download to her laptop only those emails tagged with keyword "urgent".

As another application, Waters et al. [WBDS04] show how PEKS schemes can be used to let an untrusted logging device maintain an encrypted audit log of privacy-sensitive data (e.g. user actions on a computer system) that is efficiently searchable by authorized auditors only. The entries in the audit log are encrypted under the public key of a PEKS scheme, of which the corresponding secret key is unknown to the logging device. If the device is ever confiscated, or if the logbook leaks, privacy of users and their actions is maintained. The secret key is known only to a trusted audit escrow agent, who provides (less trusted) authorized investigators with trapdoors for the keywords they want to search for.

In this paper, we investigate some consistency-related issues and results of PEKS schemes, then consider the connection to anonymous identity-based encryption (IBE), and finally discuss some new extensions.

F.1.1 Consistency in PEKS

Any cryptographic primitive must meet two conditions. One is of course a security condition. The other, which we will here call a *consistency* condition, ensures that the primitive fulfills its function. For example, for public-key encryption, the security condition is privacy. (This could be formalized in many ways, eg. IND-CPA or IND-CCA.) The consistency condition is that decryption reverses encryption, meaning that if M is encrypted under public key pk to result in ciphertext C, then decrypting C under the secret key corresponding to pk results in M being returned.

PEKS. In a PEKS scheme, Alice can provide a *gateway* with a trapdoor t_w (computed as a function of her secret key) for any keyword w of her choice. A sender encrypts a keyword w' under Alice's public key pk to obtain a ciphertext C that is sent to the gateway. The latter can apply a test function Test to t_w, C to get back 0 or 1. The consistency condition as per [BDOP04] is that if $w = w'$ then $\mathsf{Test}(t_w, C)$ returns 1 and if $w \neq w'$ it returns 0. The security condition is that the gateway learn nothing about w' beyond whether or not it equals w. (The corresponding formal notion will be denoted PEKS-IND-CPA.) The application setting is that C can be attached to an email (ordinarily encrypted for Alice under a different public key), allowing the gateway to route the email to different locations (eg. Alice's desktop, laptop or pager) based on w while preserving privacy of the latter to the largest extent possible.

Consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$. It is easy to see (cf. Proposition F.3.1) that the main construction of [BDOP04] (a random oracle (RO) model, bilinear Diffie-Hellman (BDH) based PEKS-IND-CPA secure PEKS scheme that we call $\mathcal{BDOP}$-$\mathcal{PEKS}$) fails to meet the consistency condition defined in [BDOP04] and stated above. (Specifically, there are distinct keywords w, w'

such that $\mathsf{Test}(t_w, C) = 1$ for any C that encrypts w'.) The potential problem this raises in practice is that email will be incorrectly routed.

New notions of consistency. It is natural to ask if $\mathcal{BDOP}$-$\mathcal{PEKS}$ meets some consistency condition that is weaker than theirs but still adequate in practice. To answer this, we provide some new definitions. Somewhat unusually for a consistency condition, we formulate consistency more like a security condition, via an experiment involving an adversary. The difference is that this adversary is not very "adversarial": it is supposed to reflect some kind of worst case but not malicious behavior. However this turns out to be a difficult line to draw, definitionally, so that some subtle issues arise. One outcome of this approach is that it naturally gives rise to a hierarchy of notions of consistency, namely perfect, statistical and computational. The first asks that the advantage of any (even computationally unbounded) adversary be zero; the second that the advantage of any (even computationally unbounded) adversary be negligible; the third that the advantage of any polynomial-time adversary be negligible. We note that perfect consistency as per our definition coincides with consistency as per [BDOP04], and so our notions can be viewed as natural weakenings of theirs.

An analogy. There is a natural notion of *decryption error* for encryption schemes [Gol04, Section 5.1.2]. A perfectly consistent PEKS is the analog of an encryption scheme with zero decryption error (the usual requirement). A statistically consistent PEKS is the analog of an encryption scheme with negligible decryption error (a less common but still often used condition [AD97, DNR04]). However, computational consistency is a non-standard relaxation, for consistency conditions are typically not computational. This is not because one cannot define them that way (one could certainly define a computational consistency requirement for encryption) but rather because there has never been any motivation to do so. What makes PEKS different, as emerges from the results below, is that computational consistency is relevant and arises naturally.

Consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$, revisited. The counter-example (cf. Proposition F.3.1) showing that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is not perfectly consistent extends to show that it is not statistically consistent either. However, we show (cf. Theorem F.3.3) that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is computationally consistent. In the random-oracle model, this is not under any computational assumption: the limitation on the running time of the adversary is relevant because it limits the number of queries the adversary can make to the random oracle. When the random oracle is instantiated via a hash function, we would need to assume collision-resistance of the hash function. The implication of this result is that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is probably fine to use in practice, in that incorrect routing of email, while possible in principle, is unlikely to actually happen.

A statistically consistent PEKS scheme. We provide the first construction of a PEKS scheme that is *statistically* consistent. The scheme is in the random oracle model, and is also PEKS-IND-CPA secure assuming the BDH problem is hard. The motivation for the new scheme was largely theoretical. From a foundational perspective, we wanted to know whether PEKS was an anomaly in the sense that only computational consistency is possible, or whether, like other primitives, statistical consistency could be achieved. However, it is also true that while computational consistency is arguably enough in an application, statistical might be preferable because the guarantee is unconditional.

F.1.2 PEKS and anonymous IBE

$\mathcal{BDOP}$-$\mathcal{PEKS}$ is based on the Boneh-Franklin IBE ($\mathcal{BF}$-$\mathcal{IBE}$) scheme [BF03]. It is natural to ask whether one might, more generally, build PEKS schemes from IBE schemes in some blackbox way. To this end, a transform of an IBE scheme into a PEKS scheme is suggested in [BDOP04].

Interestingly, they note that the property of the IBE scheme that appears necessary to provide PEKS-IND-CPA of the PEKS scheme is not the usual IBE-IND-CPA but rather anonymity. (An IBE scheme is anonymous if a ciphertext does not reveal the identity of the recipient [BBDP01].) While [BDOP04] stops short of stating and proving a formal result here, it is not hard to verify that their intuition is correct. Namely one can show that if the starting IBE scheme $\mathcal{IBE}$ meets an appropriate formal notion of anonymity (IBE-ANO-CPA, cf. Section F.4.1) then $\mathcal{PEKS} = \mathsf{ibe\text{-}2\text{-}peks}(\mathcal{IBE})$ is PEKS-IND-CPA, where ibe-2-peks denotes the transform suggested in [BDOP04].

Consistency in ibe-2-peks. Unfortunately, we show (cf. Theorem F.4.1) that there are IBE schemes for which the PEKS scheme outputted by ibe-2-peks is not even computationally consistent. This means that ibe-2-peks is not in general a suitable way to turn an IBE scheme into a PEKS scheme. (Although it might be in some cases, and in particular is when the starting IBE scheme is $\mathcal{BF}$-$\mathcal{IBE}$, for in that case the resulting PEKS scheme is $\mathcal{BDOP}$-$\mathcal{PEKS}$.)

new-ibe-2-peks. We propose a randomized variant of the ibe-2-peks transform that we call new-ibe-2-peks, and prove that if an IBE scheme $\mathcal{IBE}$ is IBE-ANO-CPA and IBE-IND-CPA then the PEKS scheme $\mathsf{new\text{-}ibe\text{-}2\text{-}peks}(\mathcal{IBE})$ is PEKS-IND-CPA and computationally consistent (cf. Section F.4.3). We do not know of a transform where the resulting PEKS scheme is statistically or perfectly consistent.

Anonymous IBE schemes. The above motivates finding anonymous IBE schemes. Towards this, we begin by extending Halevi's condition for anonymity [Hal05] to the IBE setting (cf. Section F.4.4). Based on this, we are able to give a simple proof that the (random-oracle model) $\mathcal{BF}$-$\mathcal{IBE}$ scheme [BF03] is IBE-ANO-CPA assuming the BDH problem is hard (cf. Theorem F.4.4). (We clarify that a proof of this result is implicit in the proof of security of the $\mathcal{BF}$-$\mathcal{IBE}$ based $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme given in [BDOP04]. Our contribution is to have stated the formal definition of anonymity and provided a simpler proof via the extension of Halevi's condition.) Towards answering the question of whether there exist anonymous IBE schemes in the standard (as opposed to random oracle) model, we present in Appendix F.8.1 an attack to show that Water's IBE scheme [Wat05] is not IBE-ANO-CPA.

F.1.3 Extensions

Anonymous HIBE. We provide definitions of anonymity for hierarchical IBE (HIBE) schemes. Our definition can be parameterized by a level, so that we can talk of a HIBE that is anonymous at level l. We note that the HIBE schemes of [GS02, BBG05] are not anonymous, even at level 1. (That of [HL02] appears to be anonymous at both levels 1 and 2 but is very limited in nature and thus turns out not to be useful for our applications.) We modify the construction of Gentry and Silverberg [GS02] to obtain a HIBE that is (HIBE-IND-CPA and) anonymous at level 1. The construction is in the random oracle model and assumes BDH is hard.

PETKS. In a PEKS scheme, once the gateway has the trapdoor for a certain keyword, it can test whether this keyword was present in any past ciphertexts or future ciphertexts. It may be useful to limit the period in which the trapdoor can be used. Here we propose an extension of PEKS that we call public-key encryption with temporary keyword search (PETKS) that allows this. A trapdoor here is created for a time interval $[s, e]$ and will only allow the gateway to test whether ciphertexts created in this time interval contain the keyword. We provide definitions of privacy and consistency for PETKS, and then show how to implement it with overhead that is only logarithmic in the total number of time periods. Our construction can use any HIBE that is anonymous at level 1. Using the above-mentioned HIBE we get a particular instantiation that is secure in the random-oracle model if BDH is hard.

IBEKS. We define the notion of an identity-based encryption with keyword search scheme. This is just like a PEKS scheme except that encryption is performed given only the identity of the receiver and a master public-key, just like in an IBE scheme. We show how to implement IBEKS given any level-2 anonymous HIBE scheme. The first suitable implementation of the latter primitive was proposed in subsequent work by Boyen and Waters [BW06].

F.1.4 Remarks

peks-2-ibe. Boneh et. al. [BDOP04] showed how to transform a PEKS-IND-CPA PEKS scheme into an IBE-IND-CPA IBE scheme. We remark that their transform requires the starting PEKS scheme to be perfectly consistent. Unfortunately, no perfectly consistent PEKS schemes are known to date. If it is only statistically or computationally consistent, the resulting IBE scheme will only meet a corresponding statistical or computational relaxation of the consistency condition for IBE schemes. Thus, the resulting scheme will not be an IBE scheme as per the standard definition of the latter [BF03].

Limited PEKS schemes. Boneh et. al. [BDOP04] also present a couple of PEKS schemes that avoid the RO model but are what they call *limited*. Both use a standard public-key encryption scheme as a building block. In the first scheme, the public key has size polynomial in the number of keywords that can be used. In the second scheme, the key and ciphertext have size polynomial in the number of trapdoors that can be securely issued to the gateway. Although these schemes are not very interesting due to their limited nature, one could ask about their consistency. In [ABC+05b], we extend our definitions of consistency to this limited setting. Interestingly, we show that based on only a computational assumption about the underlying standard public-key encryption scheme (namely, that it is IND-CPA, or even just one-way), the first scheme is *statistically* consistent. We also show that the second scheme is computationally consistent under the same assumption on the standard public-key encryption scheme, and present a variant that is statistically consistent.

Consistency of other searchable encryption schemes. Of the other papers on searchable encryption of which we are aware [SWP00, Goh03, GSW04, WBDS04], none formally define or rigorously address the notion of consistency for their respective types of searchable encryption schemes. Goh [Goh03] and Golle, Staddon, and Waters [GSW04] define consistency conditions analogous to BDOP's "perfect consistency" condition, but none of the constructions in [Goh03, GSW04] satisfy their respective perfect consistency condition. Song, Wagner, and Perrig [SWP00] and Waters et al. [WBDS04] do not formally state and prove consistency conditions for their respective searchable encryption schemes, but they, as well as Goh [Goh03], do acknowledge and informally bound the non-zero probability of a false positive.

Subsequent work. In a preliminary version of our work, we raised various open problems that have subsequently been solved. The first one of these problems was to find a construction of an (IBE-IND-CPA and) IBE-ANO-CPA IBE scheme with a proof of security in the standard model (i.e., without random oracles). This problem was solved independently by Gentry [Gen06] and by Boyen and Waters [BW06]. As a result, one can also obtain a PEKS-IND-CPA and computationally consistent PEKS scheme in the standard model due to Theorem F.4.2.

Another interesting question that we raised was to find a HIBE scheme providing anonymity at the second level, even in the RO model. This open problem was solved by Boyen and Waters [BW06], who proposed a fully anonymous HIBE scheme in the standard model.

Finally, we raised the issue of building a searchable encryption scheme that allows for more advanced searching tools such as searches for simple boolean formulas on keywords (say $w_1 \wedge w_2 \vee w_3$). First steps in this direction have been taken [GSW04, PKL04, BW07] by schemes that allow for conjunctive combinations of keywords, range queries, and subset queries.

F.2 Some definitions

NOTATION AND CONVENTIONS. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. The empty string is denoted ε. Constructs in the RO model [BR93] might use multiple random oracles, but since one can always obtain these from a single one [BR93], formal definitions will assume just one RO. Unless otherwise indicated, an algorithm may be randomized. "PT" stands for polynomial time and "PTA" for polynomial-time algorithm or adversary. We denote by $\mathbb{N}$ the set of positive integers, and by $k \in \mathbb{N}$ the security parameter. A function $\nu : \mathbb{N} \to [0,1]$ is said to be *negligible* if for every $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $\nu(k) \leq k^{-c}$ for all $k > k_c$, and it is said to be *overwhelming* if the function $|1 - \nu(k)|$ is negligible. A *message space* MsgSp is a map, assigning to every $k \in \mathbb{N}$ a set of strings, such that $\{0,1\}^k \subseteq \mathsf{MsgSp}(k) \subseteq \{0,1\}^*$ for all $k \in \mathbb{N}$ and the following conditions hold: first, there is a PTA that on input $1^k, M$ returns 1 if $M \in \mathsf{MsgSp}(k)$ and 0 otherwise; second, $\{0,1\}^{|M|} \subseteq \mathsf{MsgSp}(k)$ for all $k \in \mathbb{N}$ and $M \in \mathsf{MsgSp}(k)$.

PEKS. A *public key encryption with keyword search* (PEKS) scheme [BDOP04] $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ consists of PTAs. Via $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, where $k \in \mathbb{N}$ is the security parameter and KG is the randomized key-generation algorithm, the receiver produces its keys; via $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w)$ a sender encrypts a keyword w to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w)$ the receiver computes a trapdoor t_w for keyword w and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$ the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". Here H is a random oracle whose domain and/or range might depend on k and pk.

CONSISTENCY. The requirement of [BDOP04] can be divided into two parts. The first, which we call *right keyword consistency*, is that $\mathsf{Test}(t_w, C)$ always accepts when C encrypts w. More formally, for all $k \in \mathbb{N}$ and all $w \in \{0,1\}^*$,

$$\Pr\left[\, \mathsf{Test}^H(\mathsf{Td}^H(sk, w), \mathsf{PEKS}^H(pk, w)) = 1 \,\right] \;=\; 1\,,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. Since we will always require this, it is convenient henceforth to take it as an integral part of the PEKS notion and not mention it again, reserving the term "consistency" to only refer to what happens when the ciphertext encrypts a keyword different from the one for which the gateway is testing. In this regard, the requirement of [BDOP04], which we will call *perfect consistency*, is that $\mathsf{Test}(t_{w'}, C)$ always reject when C doesn't encrypt w'. More formally, for all $k \in \mathbb{N}$ and all distinct $w, w' \in \{0,1\}^*$,

$$\Pr\left[\, \mathsf{Test}^H(\mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1 \,\right] \;=\; 0\,,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. (We note that [BDOP04] provide informal rather than formal statements, but it is hard to interpret them in any way other than what we have done.)

PRIVACY. Privacy for a PEKS scheme [BDOP04] asks that an adversary should not be able to distinguish between the encryption of two challenge keywords of its choice, even if it is allowed to obtain trapdoors for any non-challenge keywords. Formally, we associate to an adversary $\mathcal{A}$ and a bit $b \in \{0,1\}$ the following experiment:

Experiment $\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{A}}(k)$	Oracle $\textsc{Trapd}(w)$
$WSet \leftarrow \emptyset$; $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$	$WSet \leftarrow WSet \cup \{w\}$
pick random oracle H	$t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w)$
$(w_0, w_1, state) \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot),H}(\mathtt{find}, pk)$	return t_w
$C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w_b)$	
$b' \xleftarrow{\$} \mathcal{A}^{\textsc{Trapd}(\cdot),H}(\mathtt{guess}, C, state)$	
if $\{w_0, w_1\} \cap WSet = \emptyset$ then return b' else return 0	

The PEKS-IND-CPA-*advantage* of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-1}}_{\mathcal{PEKS},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-0}}_{\mathcal{PEKS},\mathcal{A}}(k) = 1\right].$$

A scheme $\mathcal{PEKS}$ is said to be PEKS-IND-CPA-*secure* if the above advantage is a negligible function in k for all PTAs $\mathcal{A}$.

Parameter generation algorithms and the BDH problem. All pairing based schemes will be parameterized by a *pairing parameter generator*. This is a PTA $\mathcal{G}$ that on input 1^k returns the description of an additive cyclic group $\mathbb{G}_1$ of prime order p, where $2^k < p < 2^{k+1}$, the description of a multiplicative cyclic group $\mathbb{G}_2$ of the same order, and a non-degenerate bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. See [BF03] for a description of the properties of such pairings. We use $\mathbb{G}_1^*$ to denote $\mathbb{G}_1 \setminus \{0\}$, i.e. the set of all group elements except the neutral element. We define the advantage of an adversary $\mathcal{A}$ in solving the bilinear Diffie-Hellman (BDH) problem relative to a pairing parameter generator $\mathcal{G}$ as

$$\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k) \;=\; \Pr\left[\mathcal{A}(1^k, (\mathbb{G}_1, \mathbb{G}_2, p, e), P, aP, bP, cP) = e(P,P)^{abc} \;:\; \begin{array}{r}(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)\,; \\ P \xleftarrow{\$} \mathbb{G}_1^*\,;\; a, b, c \xleftarrow{\$} \mathbb{Z}_p^*\end{array}\right].$$

We say that the BDH problem is hard relative to this generator if $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}$ is a negligible function in k for all PTAs $\mathcal{A}$.

F.3 Consistency in PEKS

We show that the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not perfectly consistent, introduce new notions of statistical and computational consistency, and show that although $\mathcal{BDOP}$-$\mathcal{PEKS}$ continues to fail the former it does meet the latter. We then provide a new PEKS scheme that is statistically consistent.

F.3.1 Perfect consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$

Figure F.1 presents the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme. It is based on a pairing parameter generator $\mathcal{G}$.

Proposition F.3.1 The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not perfectly consistent.

Proof: Since the number of possible keywords is infinite, there will certainly exist distinct keywords $w, w' \in \{0,1\}^*$ such that $H_1(w) = H_1(w')$. The trapdoors for such keywords will be the same, and so $\mathsf{Test}^{H_1,H_2}(\mathsf{Td}(sk, w), \mathsf{PEKS}^{H_1,H_2}(pk, w'))$ will always return 1. ∎

It is tempting to say that, since H_1 is a random oracle, the probability of a collision is small, and thus the above really does not matter. Whether or not this is true depends on how one wants to define consistency, which is the issue we explore next.

$\mathsf{KG}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$; $P \xleftarrow{\$} \mathbb{G}_1^*$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$; $sk \leftarrow (pk, s)$
return (pk, sk)

$\mathsf{PEKS}^{H_1,H_2}(pk, w)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $T \leftarrow e(H_1(w), sP)^r$
$C \leftarrow (rP, H_2(T))$; return C

$\mathsf{Td}^{H_1}(sk, w)$
parse sk as $(pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP), s)$
$t_w \leftarrow (pk, sH_1(w))$; return t_w

$\mathsf{Test}^{H_1,H_2}(t_w, C)$
parse t_w as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP), X)$
parse C as (U, V) ; $T \leftarrow e(X, U)$
if $V = H_2(T)$ then return 1
else return 0

Figure F.1: Algorithms constituting the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme. $\mathcal{G}$ is a pairing parameter generator and H_1: $\{0,1\}^* \to \mathbb{G}_1$ and H_2: $\mathbb{G}_2 \to \{0,1\}^k$ are random oracles.

F.3.2 New notions of consistency

We consider a possible relaxation of perfect consistency and argue that it is inadequate because it is too weak. We then motivate and present our approach and definitions.

A possible relaxation of perfect consistency. One way to obtain a relaxed definition of perfect consistency is by analogy with the definition of encryption with negligible decryption error [Gol04, Section 5.1.2]. This results in asking that there exist a negligible function $\nu(\cdot)$ such that for all k and all distinct keywords w, w',

$$\forall\, w \neq w' \;\; : \;\; \Pr\Big[\, \mathsf{Test}^H(pk, \mathsf{Td}^H(sk, w'), \mathsf{PEKS}^H(pk, w)) = 1 \,\Big] \;\leq\; \nu(k)\;, \qquad \text{(F.1)}$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above. Now, since we are fixing w, w' before taking the probability, and the latter includes the choice of H_1 in the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme, the probability that $H_1(w) = H_1(w')$ is at most 2^{-k}. Our "attack" of Proposition F.3.1 therefore no longer applies. And in fact (using the techniques of our proof of Theorem F.3.3) one can show that the BDOP scheme *does* meet the above condition. However, Equation (F.1) is in our view an incorrect definition of consistency because it does not allow w, w' to depend on public quantities related to the receiver, such as its public key, the hash functions being used, or queries to them if they are random oracles. Our claim is that, as a result, the condition is too weak to guarantee that email is correctly routed by the gateway.

Our definitions. To define consistency, we take a different approach. Namely, we imagine the existence of an adversary $\mathcal{U}$ that wants to make consistency fail. More precisely, let $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ be a PEKS scheme. We associate to an adversary $\mathcal{U}$ the following experiment:

Experiment $\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$
$(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$; pick random oracle H
$(w, w') \xleftarrow{\$} \mathcal{U}^H(pk)$; $C \xleftarrow{\$} \mathsf{PEKS}^H(pk, w)$; $t_{w'} \xleftarrow{\$} \mathsf{Td}^H(sk, w')$
if $w \neq w'$ and $\mathsf{Test}^H(t_{w'}, C) = 1$ then return 1 else return 0

We define the advantage of $\mathcal{U}$ as

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \;=\; \Pr\Big[\, \mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) = 1 \,\Big]\;,$$

where the probability is taken over all possible coin flips of all the algorithms involved, and over all possible choices of random oracle H. The scheme is said to be *perfectly consistent* if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{U}$, *statistically consistent* if it is negligible for all (computationally unrestricted) adversaries $\mathcal{U}$, and *computationally consistent*

if it is negligible for all PTAs $\mathcal{U}$. We remark that we have purposely re-used the term perfect consistency, for in fact the above notion of perfect consistency coincides with the one from [BDOP04] recalled above.

STRONGER NOTIONS? In giving the adversary $\mathcal{U}$ the public key and access to the random oracle, our definition is already quite liberal. One could however, consider an even more liberal (i.e. stronger) definition in which the adversary gets a trapdoor oracle and/or a test oracle under trapdoors for keywords of its choice. To be able to tell whether or not this would be appropriate, we must ask whether in "real-life" there could be an occasion in which the keywords chosen by a sender could depend on information provided by these oracles. Given that the answer is not cut-and-dry and since we believe that our current definition is already quite strong, we opted here not to consider these stronger variants of our definition.

F.3.3 Statistical and computational consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$

Having formally defined the statistical and computational consistency requirements for PEKS schemes, we return to evaluating the consistency of $\mathcal{BDOP}$-$\mathcal{PEKS}$. We first observe that Proposition F.3.1 extends to show:

Proposition F.3.2 The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is not statistically consistent.

Proof: Recall that in the proof of Proposition F.3.1 we show that there there exist two distinct keywords $w, w' \in \{0,1\}^*$ such that $H_1(w) = H_1(w')$, and that, for these two keywords, $\mathsf{Test}(\mathsf{Td}(sk, w'), \mathsf{PEKS}(pk, w))$ will always return 1. A computationally unbounded adversary can find two such keywords by exhaustive search. ∎

On the positive side, the following means that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is probably fine in practice:

Theorem F.3.3 The $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme is computationally consistent.

Proof: Let $\mathcal{U}$ be a PTA. Let (w, w') denote the pair of keywords that $\mathcal{U}$ returns in the consistency experiment, and assume without loss of generality that $w \neq w'$. Let $r \in \mathbb{Z}_p^*$ denote the value chosen at random by $\mathsf{PEKS}^{H_1,H_2}(pk, w)$. Let $T = e(H_1(w), sP)^r$ and let $T' = e(H_1(w'), sP)^r$. Note that $\mathcal{U}$ wins exactly when $w \neq w'$ and $H_2(T) = H_2(T')$. Let $w_1, \ldots, w_{q_1}$ be the queries of $\mathcal{U}$ to H_1 and let $WSet = \{w_1, \ldots, w_{q_1(k)}\} \cup \{w, w'\}$. Let $T_1, \ldots, T_{q_2(k)}$ be the queries of $\mathcal{U}$ to H_2 and let $TSet = \{T_1, \ldots, T_{q_2(k)}\} \cup \{T, T'\}$. Let E_1 be the event that there exist distinct $v, v' \in WSet$ such that $H_1(v) = H_1(v')$, and let E_2 be the event that there exist distinct $x, x' \in TSet$ such that $H_2(x) = H_2(x')$. If $\Pr[\cdot]$ denotes the probability in the consistency experiment, then

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \;\leq\; \Pr[E_1] + \Pr[E_2] + \Pr\left[\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{BDOP}\text{-}\mathcal{PEKS},\mathcal{U}}(k) = 1 \wedge \overline{E_1} \wedge \overline{E_2}\right]. \qquad \text{(F.2)}$$

Our definition of $\mathcal{G}$ required that $|\mathbb{G}_1| > 2^k$, and hence the first and second terms are respectively upper bounded via $(q_1+2)^2/|\mathbb{G}_1| < (q_1+2)^2/2^k$ and $(q_2+2)^2/2^k$. Now we claim that if $H_1(w) \neq H_1(w')$, then $T \neq T'$. Under this claim, the last term of Equation (F.2) is 0, since if $\overline{E_1}$ occurs, then $H_1(w) \neq H_1(w')$ and $T \neq T'$, and if $\overline{E_2}$ also occurs, then $H_2(T) \neq H_2(T')$. To justify our claim above, note that if $H_1(w) \neq H_1(w')$, then $H_1(w) = \alpha P$ and $H_1(w') = \alpha' P$ for some distinct $\alpha, \alpha' \in \mathbb{Z}_p$. Setting $g = e(P,P)^{rs}$, we can rewrite T, T' as $T = g^\alpha$ and $T' = g^{\alpha'}$. Since $e(P,P)$ is a generator of $\mathbb{G}_2$, since $\mathbb{G}_2$ is of prime order p, and since p does not divide rs, g must also be a generator of $\mathbb{G}_2$. Thus $T \neq T'$. ∎

F.3.4 A statistically consistent PEKS scheme

We present the first PEKS scheme that is (PEKS-IND-CPA and) *statistically* consistent. To define the scheme, we first introduce the function $f(k) = k^{\lg(k)}$. (Any function that is super-

$\mathsf{KG}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$; $P \xleftarrow{\$} \mathbb{G}_1^*$
$s \xleftarrow{\$} \mathbb{Z}_p^*$; $pk \leftarrow (1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e)$
$sk \leftarrow (pk, s)$; return (pk, sk)

$\mathsf{PEKS}^{H_1,H_2,H_3,H_4}(pk, w)$
parse pk as $(1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e)$
if $|w| \geq f(k)$ then return w
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $T \leftarrow e(sP, H_1(w))^r$
$K_1 \leftarrow H_4(T)$; $K_2 \leftarrow H_2(T)$
$K \xleftarrow{\$} \{0,1\}^k$; $c \leftarrow K_1 \oplus K$
$t \leftarrow H_3(K||w)$
return (rP, c, t, K_2)

$\mathsf{Td}^{H_1}(sk, w)$
parse sk as $(pk = (1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e), s)$
$t_w \leftarrow (pk, sH_1(w), w)$
return t_w

$\mathsf{Test}^{H_1,H_2,H_3,H_4}(t_w, C)$
parse t_w as $((1^k, P, sP, \mathbb{G}_1, \mathbb{G}_2, p, e), sH_1(w), w)$
if $|w| \geq f(k)$ then
  if $C = w$ then return 1 else return 0
if C cannot be parsed as (rP, c, t, K_2) then return 0
$T \leftarrow e(rP, sH_1(w))$
$K \leftarrow c \oplus H_4(T)$
if $K_2 \neq H_2(T)$ then return 0
if $t = H_3(K||w)$ then return 1 else return 0

Figure F.2: Algorithms constituting the PEKS scheme $\mathcal{PEKS}$-$\mathcal{STAT}$. Here $f(k) = k^{\lg(k)}$, $\mathcal{G}$ is a pairing parameter generator and H_1: $\{0,1\}^* \to \mathbb{G}_1$, H_2: $\mathbb{G}_2 \to \{0,1\}^{3k}$, H_3: $\{0,1\}^* \to \{0,1\}^k$, and H_4: $\{0,1\}^* \to \{0,1\}^k$ are random oracles.

polynomial but sub-exponential would suffice. This choice is made for concreteness.) The algorithms constituting our scheme $\mathcal{PEKS}$-$\mathcal{STAT}$ are then depicted in Figure F.2.

The scheme uses ideas from the $\mathcal{BDOP}$-$\mathcal{PEKS}$ scheme [BDOP04] as well as from the $\mathcal{BF}$-$\mathcal{IBE}$ scheme [BF03], but adds some new elements. Note that the encryption algorithm is trivial, returning the keyword as the ciphertext, when the keyword has length more than $f(k)$. If not, the processing is more complex, depending on some random choices and numerous random oracles. In particular the random choice of "session" key K, and the fact that the random oracle H_2 is length-increasing, are important.

The first thing we stress about the scheme is that the algorithms are PT. This is because PT means in the length of the inputs, and the input of (say) the encryption algorithm includes w as well as 1^k, so it can test whether $|w| \geq f(k)$ in polynomial time. Now the following says that the scheme is private:

Proposition F.3.4 The $\mathcal{PEKS}$-$\mathcal{STAT}$ scheme is PEKS-IND-CPA-secure assuming that the BDH problem is hard relative to generator $\mathcal{G}$.

Before providing the proof, let us give some intuition. While sending w in the clear looks at first glance like it violates privacy, the reason it does not is that this only happens when w has length at least $f(k)$, and the privacy adversary is poly(k) time and thus cannot even write down such a keyword in order to query it to its challenge oracle. (This is where we use the fact that $f(k)$ is super-polynomial. We will use the fact that it is sub-exponential in the proof of statistical consistency.) The privacy adversary is thus effectively restricted to attacking the scheme only on keywords of size at most $f(k)$. Here, privacy can be reduced to solving the BDH problem using techniques used to prove IBE-IND-CPA of the $\mathcal{BF}$-$\mathcal{IBE}$ scheme [BF03] and to prove anonymity of the same scheme (cf. Theorem F.4.4).

Proof of Proposition F.3.4: Let $\mathcal{B}$ be a PTA attacking the PEKS-IND-CPA security $\mathcal{PEKS}$-$\mathcal{STAT}$ $= (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$. Say it makes at most q queries to its $\textsc{Trapd}(\cdot,\cdot)$ oracle and at most q_i queries to H_i for $i = 1, 2, 3$. (These are actually functions of k, but we drop the argument to simplify notation.) We construct a PTA $\mathcal{A}$ attacking the BDH relative to $\mathcal{G}$ such that

$$\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k) \geq \frac{1}{e(1+q)\cdot(q_2+q_4)} \cdot \left(\frac{1}{2} \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - \frac{q_3}{2^k}\right) . \qquad \text{(F.3)}$$

Adversary $\mathcal{A}(1^k, (\mathbb{G}_1, \mathbb{G}_2, p, e), P, sP, rP, \alpha P)$
$pk \leftarrow (1^k, P, sP, p, G_1, G_2, e)$; $\mathcal{Q} \leftarrow \emptyset$
$(w_0, w_1, state) \xleftarrow{\$} \mathcal{B}^{\text{TRAPD}(\cdot), H_1, H_2, H_3, H_4}(\texttt{find}, pk)$
$b \xleftarrow{\$} \{0,1\}$; $h \leftarrow H_1(w_b)$
if $d[w_b] = 0$ then abort
$K_1 \xleftarrow{\$} \{0,1\}^k$; $K_2 \xleftarrow{\$} \{0,1\}^{3k}$
$K \xleftarrow{\$} \{0,1\}^k$; $c \leftarrow K_1 \oplus K$; $t \leftarrow H_3(K \| w_b)$
$C \leftarrow (rP, c, t, K_2)$
$b' \xleftarrow{\$} \mathcal{B}^{\text{TRAPD}(\cdot), H_1, H_2, H_3, H_4}(\texttt{guess}, C, state)$
if $\mathcal{Q} \neq \emptyset$ then $T \xleftarrow{\$} \mathcal{Q}$ else abort
return $T^{x[w_b]^{-1}}$

Oracle $\text{TRAPD}(w)$
$h \leftarrow H_1(w)$
if $d[w] = 1$ then abort
$t_w \leftarrow (x[w] \cdot sP, w)$
return t_w

Oracle $H_1(w)$
if $h_1[w]$ is not defined then
flip biased coin $d[w] \in \{0,1\}$ such that $\Pr[d[w] = 1] = \delta$
$x[w] \xleftarrow{\$} \mathbb{Z}_p$
if $d[w] = 0$ then define $h_1[w] \leftarrow x[w] \cdot P$
else define $h_1[w] \leftarrow x[w] \cdot \alpha P$
return $h_1[w]$

Oracle $H_2(T)$
if $h_2[T]$ is not defined then
define $h_2[T] \xleftarrow{\$} \{0,1\}^{3k}$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{T\}$
return $h_2[T]$

Oracle $H_3(X)$
if $h_3[X]$ is not defined then
define $h_3[X] \xleftarrow{\$} \{0,1\}^k$
return $h_3[X]$

Oracle $H_4(T)$
if $h_4[T]$ is not defined then
define $h_4[T] \xleftarrow{\$} \{0,1\}^k$
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{T\}$
return $h_4[T]$

Figure F.3: Adversary $\mathcal{A}$ attacking the BDH problem.

Our adversary $\mathcal{A}$ is shown in Figure F.3. We show that $\mathcal{A}$ outputs the correct answer $T = e(P,P)^{rs\alpha}$ with probability at least the quantity on the right-hand-side of Equation (F.3).

Let $t(k)$ be a polynomial which bounds the running time of $\mathcal{B}$. So there is an integer N such that $t(k) < f(k)$ for all $k \geq N$. Notice that the PEKS algorithm of the PEKS scheme in Figure F.2 returns w in the clear when $|w| \geq f(k)$. However, the keywords output by $\mathcal{B}$ in the `find` stage have length at most $t(k)$, so if $k \geq N$, the encryption is done by the code for the case $|w| < f(k)$ shown in PEKS. Since it suffices to prove Equation (F.3) for all $k \geq N$, we assume that the encryption is done by the code for the case $|w| < f(k)$ shown in PEKS.

Let $\Pr_1[\cdot]$ denote the probability over the experiment for $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k)$ as defined in Section F.2. Let E_1 denote the event in this experiment that $\mathcal{A}$ aborts in simulating the trapdoor oracle. Let E_2 denote the event that $d[w_b] = 0$ (which also causes $\mathcal{A}$ to abort). Let E_3 denote the event that $\mathcal{Q} = \emptyset$ (which also causes $\mathcal{A}$ to abort). Let E_4 denote the event that $\mathcal{B}$ issues a query $H_2(e(rP, sH_1(w_b)))$ or $H_4(e(rP, sH_1(w_b)))$. Let $\Pr_2[\cdot]$ denote the probability over $\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}$ for a random choice for $b \in \{0,1\}$, and let b' denote the output of $\mathcal{B}$ in this experiment. Let E_5 be the event that $\mathcal{B}$ issues a query $H_2(e(rP, sH_1(w_b)))$ or $H_4(e(rP, sH_1(w_b)))$ to its oracles in this experiment. Let E_6 denote the event that $\mathcal{B}$ issues a query $K \| w_b$ to its oracle H_3, where K is the random k-bit string that $\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}$ used in PEKS when replying $\mathcal{B}$'s challenge after the find stage. Equation (F.3) follows from the following claims.

CLAIM 1. $\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{A}}(k) \geq \Pr_1[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4]/(q_2 + q_4)$.

In the above simulation if none of the events E_1, E_2 and E_3 happens, then $\mathcal{A}$ will randomly choose an element $T \xleftarrow{\$} \mathcal{Q}$ and return $T^{x[w_b]^{-1}}$. However, by definition of event E_4, one of the

elements in $\mathcal{Q}$ is equal to $e(P,P)^{sr\alpha \cdot x[w_b]}$, thus $\mathcal{A}$ has at least the probability of $1/|\mathcal{Q}| \geq 1/(q_2+q_4)$ to give the correct answer to the BDH problem. □

CLAIM 2. $\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4] = \Pr[E_4 | \neg E_1 \wedge \neg E_2] \cdot \Pr[\neg E_1 \wedge \neg E_2]$.

Notice that when event E_4 happens, the set $\mathcal{Q}$ must contain at least one element, thus E_3 is always false. Therefore we have $\Pr_1[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge E_4] = \Pr_1[\neg E_1 \wedge \neg E_2 \wedge E_4]$. The claim follows by conditioning off of the event $\neg E_1 \wedge \neg E_2$. □

CLAIM 3. $\Pr_1[E_4 \mid \neg E_1 \wedge \neg E_2] = \Pr_2[E_5]$.

Under the condition that $\mathcal{A}$ does not abort, the simulation is perfect, i.e. all $\mathcal{A}$'s answers to the simulated oracles $\textsc{Trapd}(sk,\cdot)$, $H_1(\cdot) \ldots H_4(\cdot)$ have exactly the same distribution as those in the real PEKS-IND-CPA experiment. □

CLAIM 4. $\Pr_2[E_5] \geq 1/2 \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - q_3 \cdot 2^{-k}$.

First observe that

$$\begin{aligned}
\Pr_2[b = b'] &= \Pr_2[b = b' \wedge E_5] + \Pr_2[b = b' \wedge \neg E_5 \wedge E_6] + \Pr_2[b = b' \wedge \neg E_5 \wedge \neg E_6] \\
&\leq \Pr_2[E_5] + \Pr_2[E_6] + \Pr_2[b = b' \mid \neg E_5 \wedge \neg E_6] \cdot \Pr_2[\neg E_5 \wedge \neg E_6] \\
&\leq \Pr_2[E_5] + q_3 \cdot 2^{-k} + \Pr_2[b = b' \mid \neg E_5 \wedge \neg E_6] \cdot \Pr_2[\neg E_5 \wedge \neg E_6] \qquad \text{(F.4)} \\
&\leq \Pr_2[E_5] + q_3 \cdot 2^{-k} + 1/2\,. \qquad \text{(F.5)}
\end{aligned}$$

Equation (F.4) comes from the fact that, by assumption, $\mathcal{B}$ makes at most q_3 queries to H_3. Equation (F.5) comes from the fact that, if E_5 and E_6 both do not occur, $\mathcal{B}$ learns no information from the ciphertext. Rearranging gives

$$\Pr_2[E_5] \;\geq\; \Pr_2[b = b'] - 1/2 - q_3 \cdot 2^{-k} \;=\; 1/2 \cdot \mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) - q_3 \cdot 2^{-k}\,.$$

The last equality follows from the standard result that $\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) = 2 \cdot \Pr_2[b = b'] - 1$. □

CLAIM 5. $\Pr[\neg E_1 \wedge \neg E_2] \geq 1/(e(q+1))$ for $\delta = 1/(q+1)$.

Since for every keyword w the biased coin $d[w]$ is flipped independently, and $\Pr[d[w] = 1] = \delta$ for all w, let $\mathcal{Q}_T$ be the set of queries issued by $\mathcal{B}$ to the $\textsc{Trapd}(sk,\cdot)$ oracle, then

$$\Pr[\neg E_1 \wedge \neg E_2] = \delta \cdot \prod_{w \in \mathcal{Q}_T} (1-\delta) = \delta \cdot (1-\delta)^{|\mathcal{Q}_T|} \geq \delta \cdot (1-\delta)^q$$

The last quantity is maximized at $\delta = 1/(q+1)$ with value at least $1/e(q+1)$. ▌

Let us move to the more interesting claim, namely consistency:

Proposition F.3.5 The $\mathcal{PEKS}$-$\mathcal{STAT}$ scheme is statistically consistent.

Before providing the proof, let us give some intuition. The main issue is that the computationally unbounded consistency adversary $\mathcal{U}$ can easily find any collisions that exist for the random-oracle hash functions. Let w, w' denote the keywords output by the adversary $\mathcal{U}$. We proceed via a case analysis. One can show that if either w or w' have length at least $f(k)$ then Test will not be wrong. The interesting case is when w, w' both have length at most $f(k)$. Let (rP, c, t, K_2) denote the challenge ciphertext formed by encrypting w. Let $T = e(rP, H_1(w))$ and let $K = c \oplus H_4(T)$ be the underlying session key. Let $T' = e(rP, H_1(w'))$ and let $K' = c \oplus H_4(T')$. Now consider two cases.

The first case is that $H_1(w) \neq H_1(w')$. Properties of pairings imply $T \neq T'$. Now we claim that this means $K_2 = H_2(T) \neq H_2(T')$ with high probability, and thus Test will correctly reject,

meaning $\mathcal{U}$ does not win. This is *not* merely because H_2 is random, for remember the adversary is not computationally bounded and can search for, and find, any collisions that exist. The reason is that H_2 is with high probability an injective function and collisions for it simply do not exist. The reason for this is that its domain is $\mathbb{G}_2$ which has size $p < 2^{k+1}$ (our definition of a pairing parameter generator required this) but H_2 outputs $3k$ bits, and thus a union bound can be used to show that H_2 is injective except with probability $4 \cdot 2^{-k}$.

The second case, which is the harder one, is that $H_1(w) = H_1(w')$ (again, we cannot prevent $\mathcal{U}$ from finding collisions in H_1), and this is where we will use the fact that $f(k)$ is sub-exponential. Here the idea is that at the time it chooses w, w', adversary $\mathcal{U}$ does not know the value of the session key K that is randomly chosen later. We divide pairs (V, V') of strings of length at most $f(k)$ (candidate keywords) into two classes. A pair is *heavy* if there are "lots" of session keys L such that $H_3(L \,\|\, V) = H_3(L \,\|\, V')$, and *light* otherwise, where "lots" is defined as $2^{k/2}$. Now we again consider two cases. If (w, w') is light then the randomly chosen K has only a $2^{-k/2}$ chance of being a session key for which $H_3(K \,\|\, w) = H_3(K \,\|\, w')$ and thus Test will most likely reject, so $\mathcal{U}$ does not win. Next we use an occupancy problem based counting argument to show that the probability (over H_3) that a particular pair (V, V') of keywords is heavy is *double* exponentially small in k. But the number of choices of keyword pairs is $2^{O(f(k))}$ which is sub-double-exponentially small by choice of $f(k)$, and thus a union bound allows us to conclude that (w, w') is not likely to be heavy.

Proof of Proposition F.3.5: Let $\mathcal{U}$ be a computationally unbounded adversary algorithm. We show that there is a constant $c > 0$ such that

$$\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \leq O(2^{-ck}) \; .$$

Consider the experiment $\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k)$. Let w, w' denote the keywords output by $\mathcal{U}$ and assume they are distinct, since otherwise $\mathcal{U}$ does not win. Let $\textsc{Win}$ be the event that the experiment outputs 1. Let $r, \mathbf{K}$ be the random choices made by $\mathsf{PEKS}^{H_1,H_2,H_3,H_4}(pk, w)$ in the experiment. Then we let

$$\begin{array}{rclcrcl} \mathbf{T} & = & e(rP, sH_1(w)) & & \mathbf{T}' & = & e(rP, sH_1(w')) \\ \mathbf{c} & = & \mathbf{K} \oplus H_4(\mathbf{T}) & & \mathbf{K}' & = & \mathbf{c} \oplus H_4(\mathbf{T}') \\ \mathbf{K}_2 & = & H_2(\mathbf{T}) & & \mathbf{K}'_2 & = & H_2(\mathbf{T}') \\ \mathbf{t} & = & H_3(\mathbf{K} \,\|\, w) & & \mathbf{t}' & = & H_3(\mathbf{K}' \,\|\, w') \; . \end{array}$$

The random choices of H_1, H_2, H_3, H_4, r and $\mathbf{K}$ determine all these random variables. Let $\textsc{Bad}$ be the event that $\mathsf{Test}(t_{w'}, (\mathbf{C}, \mathbf{c}, \mathbf{t}, \mathbf{K}_2)) = 1$. Let $\textsc{Big}$ be the event that either w or w' has length greater than or equal to $f(k)$. Then

$$\begin{array}{rcl} \mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) & = & \Pr[\,\textsc{Bad}\,] \;\leq\; \Pr[\,\textsc{Bad} \wedge \textsc{Big}\,] + \Pr[\,\textsc{Bad} \wedge \neg\textsc{Big}\,] \\ & \leq & \Pr[\,\textsc{Bad} \mid \textsc{Big}\,] + \Pr[\,\textsc{Bad} \wedge \neg\textsc{Big}\,] \; . \end{array}$$

Suppose $\textsc{Big}$ holds. If $|w'| \geq f(k)$ then $\mathsf{Test}(t_{w'}, \mathbf{C})$ will return 1 only if $\mathbf{C} = w'$. But this will not be the case because either $|w| \geq f(k)$ and $\mathbf{C} = w \neq w'$, or $|w| < f(k)$ and, for large enough k, $|\mathbf{C}| < f(k) \leq |w'|$. On the other hand if $|w| \geq f(k)$ and $|w'| < f(k)$ then $\mathbf{C} = w$ and the latter cannot be parsed as an appropriate 4-tuple (rP, c, t, K_2), so Test will return 0. We conclude that $\Pr[\,\textsc{Bad} \mid \textsc{Big}\,] = 0$ for all large enough k. We now want to bound

$$\begin{array}{l} \Pr[\,\textsc{Bad} \wedge \neg\textsc{Big}\,] \\ \quad = \; \underbrace{\Pr[\,\textsc{Bad} \wedge \neg\textsc{Big} \wedge H_1(w) \neq H_1(w')\,]}_{p_1} + \underbrace{\Pr[\,\textsc{Bad} \wedge \neg\textsc{Big} \wedge H_1(w) = H_1(w')\,]}_{p_2} \; . \end{array}$$

We bound p_1, p_2 in turn. We let S be the set of all distinct pairs (g, g') of elements in $\mathbb{G}_1$. So p_1 is at most the sum, over all $(g, g') \in S$, of the product terms

$$\Pr\left[\, H_2(e(rP, g)) = H_2(e(rP, g')) \mid (H_1(w), H_1(w')) = (g, g') \,\right] \cdot \Pr\left[\, (H_1(w), H_1(w')) = (g, g') \,\right] .$$

Properties of pairings tell us that $g \neq g'$ implies $e(rP, g) \neq e(rP, g')$. So due to the randomness of H_2, the first term of each product above is 2^{-3k}. However, there are at most p^2 choices for the pair (g, g'), and we know that $p < 2^{k+1}$. Thus we have

$$p_1 \leq p^2 \cdot 2^{-3k} \leq 2^{2k+2-3k} = 4 \cdot 2^{-k} .$$

(As we discussed above, the intuition here is that with probability at least $1 - 4 \cdot 2^{-k}$ the function H_2 is injective.) We now proceed to bound p_2. In this argument, we regard H_1 as fixed. (Formally, imagine that we condition on a particular choice of H_1. This suffices since what follows holds for all values of this choice.) Let U be the set of all pairs (V, V') of distinct keywords of length at most $f(k)$ each such that $H_1(V) = H_1(V')$. For any $(V, V') \in U$ we let

$$\mathsf{Keys}(V, V') \;=\; \{\, A \in \{0,1\}^k \;:\; H_3(A \,\|\, V) = H_3(A \,\|\, V') \,\} .$$

We say that (V, V') is *heavy* if $|\mathsf{Keys}(V, V')| \geq 2^{k/2}$, and *light* otherwise. We let $\mathsf{Lt}(V, V')$ denote the event that (V, V') is light and $\mathsf{Hw}(V, V')$ the event that (V, V') is heavy, where the probability is over the choice of H_3 only. Then $p_2 \leq p_L + p_H$ where

$$\begin{aligned} p_L &= \sum_{(V,V') \in U} \Pr\left[\, \textsc{Bad} \wedge (w, w') = (V, V') \wedge \mathsf{Lt}(V, V') \,\right] \\ p_H &= \sum_{(V,V') \in U} \Pr\left[\, \textsc{Bad} \wedge (w, w') = (V, V') \wedge \mathsf{Hw}(V, V') \,\right] . \end{aligned}$$

We bound these in turn. We have

$$\begin{aligned} p_L &= \sum_{(V,V') \in U} \Pr\left[\, \textsc{Bad} \mid (w, w') = (V, V') \wedge \mathsf{Lt}(V, V') \,\right] \cdot \Pr\left[\, (w, w') = (V, V') \wedge \mathsf{Lt}(V, V') \,\right] \\ &\leq \sum_{(V,V') \in U} \frac{2^{k/2}}{2^k} \cdot \Pr\left[\, (w, w') = (V, V') \wedge \mathsf{Lt}(V, V') \,\right] \qquad \text{(F.6)} \\ &= 2^{-k/2} \cdot \sum_{(V,V') \in U} \Pr\left[\, (w, w') = (V, V') \wedge \mathsf{Lt}(V, V') \,\right] \\ &\leq 2^{-k/2} . \end{aligned}$$

Equation (F.6) is justified by the definition of the Test, the fact that $\mathbf{K}$ is chosen at random from $\{0,1\}^k$ and the fact that (V, V') is light. Now we turn to bounding p_H.

Claim. For any $(V, V') \in U$,

$$\Pr\left[\, \mathsf{Hw}(V, V') \,\right] \leq O(2^{-2^{k/2}}) ,$$

where the probability is only over the choice of H_3.

Note the bound of the claim is double-exponentially small. We prove the claim later. Using it we can conclude via the union bound:

$$\begin{aligned} p_H &= \sum_{(V,V') \in U} \Pr\left[\, \textsc{Bad} \wedge (w, w') = (V, V') \wedge \mathsf{Hw}(V, V') \,\right] \\ &\leq \sum_{(V,V') \in U} \Pr\left[\, \mathsf{Hw}(V, V') \,\right] \;\leq\; 2^{2+2f(k)} \cdot O(2^{-2^{k/2}}) \;\leq\; O(2^{-g(k)}) , \end{aligned}$$

where $g(k) = 2^{k/2} - 2 - 2f(k) = \Omega(2^{k/2})$. So certainly $2^{-g(k)}$ is $O(2^{-k})$.

Proof of Claim. We use an occupancy problem approach:

$$\begin{aligned}\Pr\left[\,\mathsf{Hw}(V,V')\,\right] &= \sum_{i=2^{k/2}}^{2^k} \binom{2^k}{i} \cdot (2^{-k})^i \cdot (1-2^{-k})^{2^k-i} \;\le\; \sum_{i=2^{k/2}}^{2^k} \binom{2^k}{i} \cdot (2^{-k})^i \\ &\le \sum_{i=2^{k/2}}^{2^k} \left(\frac{2^k \cdot e}{i}\right)^i \cdot (2^{-k})^i \;\le\; \sum_{i=2^{k/2}}^{2^k} \left(\frac{e}{i}\right)^i \;\le\; \sum_{i=2^{k/2}}^{\infty} \left(\frac{e}{i}\right)^i .\end{aligned}$$

Let $x = e2^{-k/2}$. For $k \ge 6$, we have $x \le 1/2$. So the above is at most

$$\sum_{i=2^{k/2}}^{\infty} x^i \;=\; x^{2^{k/2}} \cdot \sum_{i=0}^{\infty} x^i \;=\; x^{2^{k/2}} \frac{1}{1-x} \;\le\; \frac{2}{2^{2^{k/2}}} ,$$

as desired. ∎

F.4 PEKS and anonymous IBE

We formally define anonymity of IBE schemes and investigate the relation between PEKS and anonymous IBE.

F.4.1 Definitions

IBE schemes. An *identity-based encryption* (IBE) scheme [Sha85, BF03] $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ consists of four PTAs. Via $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$ the master generates master keys for security parameter $k \in \mathbb{N}$; via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$ the master computes the secret key for identity id; via $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M)$ a sender encrypts a message M to identity id to get a ciphertext; via $M \leftarrow \mathsf{Dec}^H(usk, C)$ the possessor of secret key usk decrypts ciphertext C to get back a message. Here H is a random oracle with domain and range possibly depending on k and pk. Associated to the scheme is a message space MsgSp obeying the conventions discussed in Section F.2. For consistency, we require that for all $k \in \mathbb{N}$, all identities id and messages $M \in \mathsf{MsgSp}(k)$ we have $\Pr[\mathsf{Dec}^H(\mathsf{KeyDer}^H(msk, id), \mathsf{Enc}^H(pk, id, M)) = M] = 1$, where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

Privacy and anonymity. Privacy (IBE-IND-CPA) follows [BF03] while anonymity (IBE-ANO-CPA) is a straightforward adaptation of [BBDP01] to IBE schemes. Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme with associated message space MsgSp. To an adversary $\mathcal{A}$ and bit $b \in \{0,1\}$, we associate the following experiments:

Experiment $\mathbf{Exp}^{\text{ibe-ind-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
pick random oracle H
$(id, M_0, M_1, state) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer}(\cdot),H}(\mathtt{find}, pk)$
if $\{M_0, M_1\} \not\subseteq \mathsf{MsgSp}(k)$ then return 0
$C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M_b)$
$b' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer}(\cdot),H}(\mathtt{guess}, C, state)$
if $id \notin IDSet$ and $|M_0| = |M_1|$
then return b' else return 0

Experiment $\mathbf{Exp}^{\text{ibe-ano-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
pick random oracle H
$(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer},H}(\mathtt{find}, pk)$
if $M \notin \mathsf{MsgSp}(k)$ then return 0
$C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, M)$
$b' \xleftarrow{\$} \mathcal{A}^{\textsc{KeyDer},H}(\mathtt{guess}, C, state)$
if $\{id_0, id_1\} \cap IDSet = \emptyset$
then return b' else return 0

$\overline{\mathsf{Setup}}(1^k)$
$(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
$R \xleftarrow{\$} \{0,1\}^k$
$\overline{pk} \leftarrow (pk, R)$; $\overline{msk} \leftarrow (msk, R)$
return $(\overline{pk}, \overline{msk})$

$\overline{\mathsf{KeyDer}}(\overline{msk}, id)$
parse $\overline{msk}$ as (msk, R)
$usk \xleftarrow{\$} \mathsf{KeyDer}(msk, id)$
$\overline{usk} \leftarrow (usk, R)$
return $\overline{usk}$

$\overline{\mathsf{Enc}}(\overline{pk}, id, M)$
parse $\overline{pk}$ as (pk, R)
$C \xleftarrow{\$} \mathsf{Enc}(pk, id, M\|R)$
return C

$\overline{\mathsf{Dec}}(\overline{usk}, C)$
parse $\overline{usk}$ as (usk, R)
$X \leftarrow \mathsf{Dec}(usk, C)$
parse X as $M\|R'$ where $|R'| = k$
if $R' = R$ then return M
else return 0^k

Figure F.4: IBE scheme for proof of Theorem F.4.1.

where the oracle KeyDer(id) is defined as

$$IDSet \leftarrow IDSet \cup \{id\} \; ; \; usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id) \; ; \;\; \text{Return } usk[id]$$

For prop $\in \{\text{ind}, \text{ano}\}$, we define the advantage of $\mathcal{A}$ in the corresponding experiment as

$$\mathbf{Adv}^{\text{ibe-prop-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{ibe-prop-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-prop-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right].$$

IBE scheme $\mathcal{IBE}$ is said to be IBE-IND-CPA-*secure* (resp., IBE-ANO-CPA-*secure*) if the respective advantage function is negligible for all PTAs $\mathcal{A}$.

F.4.2 The ibe-2-peks transform

The ibe-2-peks transform suggested in [BDOP04] takes input an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme (i.e., $\mathsf{KG} = \mathsf{Setup}$). The trapdoor t_w associated to keyword w is the secret key that the IBE scheme would assign to the identity w (i.e., $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$). A keyword w is PEKS-encrypted by IBE-encrypting the message 0^k for the identity w (i.e., $\mathsf{PEKS}(pk, w) = \mathsf{Enc}(pk, w, 0^k)$). Finally, testing is done by checking that the ciphertext decrypts to 0^k (i.e., $\mathsf{Test}(t_w, C)$ returns 1 iff $\mathsf{Dec}(t_w, C) = 0^k$).

We know that $\mathcal{BF}$-$\mathcal{IBE}$ is anonymous (Theorem F.4.4), that $\mathcal{BDOP}$-$\mathcal{PEKS} = \mathsf{ibe\text{-}2\text{-}peks}(\mathcal{BF}\text{-}\mathcal{IBE})$, and that $\mathcal{BDOP}$-$\mathcal{PEKS}$ is not statistically consistent (Proposition F.3.2). Thus, we can conclude that the ibe-2-peks transform does not necessarily yield a statistically consistent PEKS scheme. Unfortunately, as the following theorem shows, the ibe-2-peks transform does not necessarily yield a computationally consistent PEKS scheme either (under the minimal assumption of the existence of some IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme). As a result, ibe-2-peks is not in general a suitable way to obtain a PEKS scheme.

Theorem F.4.1 Assume there exist IBE-ANO-CPA-secure and IBE-IND-CPA-secure IBE schemes. Then there exists a IBE-ANO-CPA-secure and IBE-IND-CPA-secure IBE scheme $\overline{\mathcal{IBE}}$ such that the PEKS scheme $\mathcal{PEKS}$ derived from $\overline{\mathcal{IBE}}$ via ibe-2-peks is not computationally consistent.

Proof of s: The proof of Theorem F.4.1 is quite simple and its details are omitted here. Instead, we only provide the general intuition behind it. In order to show that ibe-2-peks does not necessarily yield a computationally consistent PEKS scheme, we first assume the existence of a IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and

then build an IBE scheme $\overline{\mathcal{IBE}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{KeyDer}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ as shown in Figure F.4. It is easy to see that the IBE-IND-CPA- and IBE-ANO-CPA-security of $\overline{\mathcal{IBE}}$ follows from simple reductions from the security of $\mathcal{IBE}$. Now, let $\mathcal{PEKS}$ denote the PEKS scheme outputted by ibe-2-peks on input $\overline{\mathcal{IBE}}$. Clearly, $\mathcal{PEKS}$ is not computationally consistent as its test algorithm outputs 1 with overwhelming probability, when given the trapdoor for the wrong keyword. The only case in which it outputs 0 when given the wrong trapdoor is when the last k bits of the decryption of the ciphertext C with the wrong trapdoor matches the random value R in the public key $\overline{pk}$, but this only happens with negligible probability due to the IBE-IND-CPA security of the IBE scheme $\mathcal{IBE}$. ∎

F.4.3 The new-ibe-2-peks transform

The negative result in Theorem F.4.1 raises the question: Does the existence of IBE schemes imply the existence of computationally consistent PEKS schemes? We answer that in the affirmative by presenting a revision of the ibe-2-peks transform, called new-ibe-2-peks, that transforms any IBE-IND-CPA- and IBE-ANO-CPA-secure IBE scheme into a PEKS-IND-CPA-secure and computationally consistent PEKS scheme. It is similar to ibe-2-peks except that instead of always using 0^k as the message encrypted, the PEKS-encryption algorithm chooses and encrypts a random message R and appends R in the clear to the ciphertext. In more detail, the new-ibe-2-peks transform takes input an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ and returns a PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows. The public key pk and secret key sk of the receiver in the PEKS scheme are the master public and secret keys, respectively, of the IBE scheme. (I.e. $\mathsf{KG} = \mathsf{Setup}$.) The trapdoor associated to keyword w is the secret key that the IBE scheme would assign to the identity w. (I.e. $\mathsf{Td}(sk, w) = \mathsf{KeyDer}(sk, w)$.) PEKS-encryption of keyword w is done as follows: $\mathsf{PEKS}(pk, w)$ picks $R \xleftarrow{\$} \{0,1\}^k$, lets $C \xleftarrow{\$} \mathsf{Enc}(pk, w, R)$, and returns (C, R) as the ciphertext. Finally, $\mathsf{Test}(t_w, (C, R))$ returns 1 iff $\mathsf{Dec}(t_w, C) = R$.

Intuitively, this construction avoids the problem of oddly-behaving Dec algorithms by making sure that the *only* way to ruin the consistency of the PEKS scheme is by correctly guessing the value encrypted by a ciphertext, using the secret key of a different identity, which should not be possible for an IBE-IND-CPA-secure IBE scheme. Hence, the consistency of the resulting PEKS scheme is due to the data privacy property of the IBE scheme, while the data privacy property of the PEKS scheme comes from the anonymity of the IBE scheme. The formal result statement and proof follow.

Theorem F.4.2 Let $\mathcal{IBE}$ be an IBE scheme and let $\mathcal{PEKS}$ be the PEKS scheme derived from $\mathcal{IBE}$ via new-ibe-2-peks. If $\mathcal{IBE}$ is IBE-IND-CPA-secure, then $\mathcal{PEKS}$ is computationally consistent. Further, if $\mathcal{IBE}$ is IBE-ANO-CPA-secure, then $\mathcal{PEKS}$ is PEKS-IND-CPA-secure.

Proof: Let $\mathcal{U}$ be any PTA attacking the computational consistency of $\mathcal{PEKS}$, and consider the following PTA $\mathcal{A}$ attacking the IBE-IND-CPA-security of $\mathcal{IBE}$. In its `find` stage, given master public key pk, adversary $\mathcal{A}$ runs $\mathcal{U}(pk)$ to get keywords w, w'. It returns w as the challenge identity and $R_0, R_1 \xleftarrow{\$} \{0,1\}^k$ as the challenge messages. In the `guess` stage, given challenge ciphertext C (that encrypts R_b under identity w for challenge bit $b \in \{0,1\}$), $\mathcal{A}$ uses its key-derivation oracle to obtain a trapdoor $t_{w'}$ for w'. If $\mathsf{Dec}(t_{w'}, C) = R_1$ then it returns 1 else it returns 0. It is easy to see that

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{ibe-ind-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k)] = 1\right] &\geq \Pr\left[\mathbf{Exp}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) = 1\right] \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ind-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k)] = 1\right] &\leq 2^{-k}\,.
\end{aligned}$$

Thus $\mathbf{Adv}^{\text{peks-consist}}_{\mathcal{PEKS},\mathcal{U}}(k) \leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}}(k) + 2^{-k}$, proving the first claim of the theorem.

Let $\mathcal{B}$ be any PTA attacking the PEKS-IND-CPA-security of $\mathcal{PEKS}$, and consider the following PTA $\mathcal{A}$ attacking the IBE-ANO-CPA-security of $\mathcal{IBE}$. In its `find` stage, given master public key pk, adversary $\mathcal{A}$ runs $\mathcal{B}(\texttt{find}, pk)$ to get challenge keywords w_0, w_1, which it returns along with a message $R \xleftarrow{\$} \{0,1\}^k$. In the `guess` stage, given challenge ciphertext C (that encrypts R under identity w_b for challenge bit $b \in \{0,1\}$), $\mathcal{A}$ runs $\mathcal{B}$, in its `guess` stage, with challenge ciphertext (C, R), to get its guess bit b', which $\mathcal{A}$ returns. In both stages, $\mathcal{A}$ answers any trapdoor-oracle queries of $\mathcal{B}$ via its key-derivation oracle. It is easy to see that for $b = 0, 1$,

$$\Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-b}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{peks-ind-cpa-b}}_{\mathcal{PEKS},\mathcal{B}}(k) = 1\right].$$

Thus $\mathbf{Adv}^{\text{peks-ind-cpa}}_{\mathcal{PEKS},\mathcal{B}}(k) \le \mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{IBE},\mathcal{A}}(k)$, proving the second claim of the theorem. ∎

F.4.4 A sufficient condition for anonymity

Halevi [Hal05] provides a simple sufficient condition for an IND-CPA public-key encryption scheme to meet the notion of anonymity (a.k.a. key-privacy) of [BBDP01]. The condition is that even a computationally unbounded adversary, given public keys pk_0, pk_1 and the encryption of a random message under pk_b, have only a negligible advantage in determining the random challenge bit b. Towards finding anonymous IBE schemes (a task motivated by Theorem F.4.2) we extend Halevi's condition to identity-based encryption. In the process we also extend it in two other ways: first to handle the random oracle model (the standard model is a special case) and second to weaken the statistical (i.e. information-theoretic) requirement of [Hal05] to a computational one. (The application of this paper does not need the last extension, but it may be useful in other contexts.)

We begin by defining a relevant (new) notion of security that we call IBE-ANO-RE-CPA. Let $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an IBE scheme with associated message space MsgSp. We associate to an adversary $\mathcal{A}$ and bit $b \in \{0,1\}$ the following experiment:

Experiment $\mathbf{Exp}^{\text{ibe-ano-re-b}}_{\mathcal{IBE},\mathcal{A}}(k)$
- $IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
- pick random oracle H
- $(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{find}, pk)$
- if $M \notin \mathsf{MsgSp}(k)$ then return 0
- $R \xleftarrow{\$} \{0,1\}^{|M|}$; $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, R)$
- $b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{guess}, C, state)$
- if $\{id_0, id_1\} \cap IDSet = \emptyset$ then return b'
- else return 0

Oracle KeyDer(id)
- $IDSet \leftarrow IDSet \cup \{id\}$
- $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$
- return $usk[id]$

The IBE-ANO-RE-CPA-*advantage* of an adversary $\mathcal{A}$ in violating the anonymity of the scheme $\mathcal{IBE}$ is defined as

$$\mathbf{Adv}^{\text{ibe-ano-re}}_{\mathcal{IBE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right].$$

A scheme $\mathcal{IBE}$ is said to be IBE-ANO-RE-CPA-*secure* if the above advantage is a negligible function in k for all PTAs $\mathcal{A}$.

Lemma F.4.3 Let $\mathcal{IBE}$ be an IBE scheme that is IBE-IND-CPA and IBE-ANO-RE-CPA-secure. Then it is also IBE-ANO-CPA-secure.

Proof of Lemma F.4.3: The proof is a simple hybrid argument. Let $\mathcal{A}$ be a PTA attacking the IBE-ANO-CPA-security of $\mathcal{IBE}$. It is easy to construct PTAs $\mathcal{A}_1, \mathcal{A}_3$ attacking the

$\mathsf{Setup}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$; $P \xleftarrow{\$} \mathbb{G}_1^*$; $s \xleftarrow{\$} \mathbb{Z}_p^*$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, sP)$; $msk \leftarrow s$
return (pk, msk)

$\mathsf{KeyDer}^{H_1}(msk, id)$
$sk[id] \leftarrow sH_1(id)$
return $sk[id]$

$\mathsf{Enc}^{H_1,H_2}(pk, id, M)$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $T \leftarrow e(H_1(id), sP)^r$
$C \leftarrow (rP, M \oplus H_2(T))$
return C

$\mathsf{Dec}^{H_2}(sk[id], C)$
parse C as (U, V)
$T \leftarrow e(sk[id], U)$; $M \leftarrow V \oplus H_2(T)$
return M

Figure F.5: Algorithms of the IBE scheme $\mathcal{BF}\text{-}\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$. Here $\mathcal{G}$ is a pairing parameter generator and H_1: $\{0,1\}^* \to \mathbb{G}_1^*$ and H_2: $\mathbb{G}_2 \to \{0,1\}^k$ are random oracles. The message space is defined by $\mathsf{MsgSp}(k) = \{0,1\}^k$ for all $k \in \mathbb{N}$.

IBE-IND-CPA-security of $\mathcal{IBE}$, and PTA $\mathcal{A}_2$ attacking the IBE-ANO-RE-CPA-security of $\mathcal{IBE}$, such that

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}_1}(k) \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-1}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ano-re}}_{\mathcal{IBE},\mathcal{A}_2}(k) \\
\Pr\left[\mathbf{Exp}^{\text{ibe-ano-re-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ibe-ano-cpa-0}}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] &\leq \mathbf{Adv}^{\text{ibe-ind-cpa}}_{\mathcal{IBE},\mathcal{A}_3}(k)\ .
\end{aligned}$$

Summing concludes the proof. We omit the details, save to remark that we use here the second convention about message spaces noted in Section F.2. ▮

F.4.5 Anonymity of $\mathcal{BF}$-$\mathcal{IBE}$

The Boneh-Franklin BasicIdent IBE scheme [BF03] is shown in Figure F.5. We apply Lemma F.4.3 to give a simple proof that it is IBE-ANO-CPA.

Theorem F.4.4 The $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme is IBE-ANO-CPA-secure assuming that the BDH is hard relative to generator $\mathcal{G}$.

Proof: Given Lemma F.4.3, and given that the $\mathcal{BF}\text{-}\mathcal{IBE}$ scheme is IBE-IND-CPA-secure [BF03], it suffices to show that the scheme is IBE-ANO-RE-CPA-secure. Notice that the ciphertext C in Figure F.5 has two parts, namely $U = rP$ and $V = M \oplus H_2(T)$. The value U is chosen uniformly at random from $\mathbb{G}_1^*$ by the encryption algorithm. If the message M is chosen uniformly at random from $\{0,1\}^k$, then V is also uniformly distributed in $\{0,1\}^k$ and independent of the $H_2(T)$. Thus in both the 0- and 1- worlds of the IBE-ANO-RE-CPA-security game, the challenge ciphertext C has exactly the same distribution. Therefore any adversary against IBE-ANO-RE-CPA-security will have 0 advantage. ▮

F.5 Anonymous HIBE

F.5.1 Definitions

HIBE SCHEMES. A *hierarchical identity-based encryption* (HIBE) scheme [HL02, GS02, BBG05] is a generalization of an IBE scheme in which an identity is a vector of strings $id = (id_1, \ldots, id_l)$ with the understanding that when $l = 0$ this is the empty vector $()$. The number of components

in this vector is called the level of the identity and is denoted $|id|$. If $0 \le i \le l$ then $id|_i = (id_1, \ldots, id_i)$ denotes the vector containing the first i components of id. If $|id'| \ge l+1$ ($l \ge 0$) and $id'|_l = id$ then we say that id is an ancestor of id', or equivalently, that id' is a descendant of id. If the level of id' is $l+1$ then id is a parent of id', or, equivalently, id' is a child of id. For any id with $|id| \ge 1$ we let $\mathrm{par}(id) = id|_{|id|-1}$ denote its parent. Two nodes $id = (id_1, \ldots, id_l)$ and $id' = (id'_1, \ldots, id'_l)$ at level l are said to be siblings iff $id|_{l-1} = id'|_{l-1}$. Moreover, if $id_l < id'_l$ in lexicographic order, then id is a left sibling of id' and id' is a right sibling of id. An identity at level one or more can be issued a secret key by its parent. (And thus an identity can issue keys for any of its descendants if necessary.)

Formally a HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ consists of four PTAs. Via $(pk, msk = usk[()]) \xleftarrow{\$} \mathsf{Setup}(1^k)$, where $k \in \mathbb{N}$ is a security parameter, the root generates master keys, with the secret key being associated to the (unique) identity () at level 0. Via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id)$ the parent of an identity id with $|id| \ge 1$ can compute a secret key for id. Note that by iteratively applying the KeyDer algorithm a user id can derive secret keys for any of its descendants id'; we occasionally use the notation $usk[id'] \xleftarrow{\$} \mathsf{KeyDer}^H(usk[id], id')$ to denote this process. Via $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M)$ a sender encrypts a message M to identity id to get a ciphertext; via $M \leftarrow \mathsf{Dec}^H(usk[id], C)$ the identity id decrypts ciphertext C to get back a message. Here H is a random oracle with domain and range possibly depending on k and pk. Associated to the scheme is a message space MsgSp obeying the conventions discussed in Section F.2. For consistency, we require that for all $k \in \mathbb{N}$, all identities id with $|id| \ge 1$ and all messages $M \in \mathsf{MsgSp}(k)$,

$$\Pr\left[\, \mathsf{Dec}^H(\mathsf{KeyDer}^H(usk[\mathrm{par}(id)], id), \mathsf{Enc}^H(pk, id, M)) = M \,\right] \;=\; 1\ ,$$

where the probability is taken over the choice of $(pk, usk[()]) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

Privacy and anonymity. The notion of privacy for HIBE schemes is analogous to that for IBE schemes (IBE-IND-CPA) but using identity vectors rather than identity strings and where the adversary is not allowed to query the KeyDer oracle for the secret key of any ancestor of the identity under attack. Since we will deal with schemes where privacy holds only up to some level, the notion is parameterized by a maximum depth function $d\colon \mathbb{N} \to \mathbb{N}$, and all identities id (in queries or challenges) must have $|id| \le d(k)$. To allow a fine-grained treatment of anonymity we introduce the concept of anonymity at a set $L(k)$ of levels, meaning that in an experiment the adversary $\mathcal{A}$ is challenged to distinguish two distinct identities differing only at levels $l \in L(k)$. (Here for each k, $L(k)$ is a finite set of integers. For ease of notation, we will write l rather than $\{l\}$ when $L(k) = \{l\}$ is a singleton set.)

Formally, let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be an identity-based encryption scheme with message space MsgSp, let $d : \mathbb{N} \to \mathbb{N}$ be the maximum depth, and let L be a set of levels. Let $\mathrm{diff}(\cdot,\cdot)$ be the function that returns the set of coordinates at which the input identities differ, and $\mathrm{anc}(\cdot)$ the function returning the set of ancestors of the input identity. To any bit $b \in \{0,1\}$ and any adversary $\mathcal{A}$, we associate the experiments:

Experiment $\mathbf{Exp}^{\text{hibe-ind-cpa-b}[d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
pick random oracle H
$(id, M_0, M_1, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{find}, pk)$
if $|M_0| \neq |M_1|$ or $|id| > d(k)$
or $\{M_0, M_1\} \not\subseteq \mathsf{MsgSp}(k)$
then return 0
$C \xleftarrow{\$} \mathsf{Enc}^H(pk, id, M_b)$
$b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{guess}, C, state)$
if $IDSet \cap \text{anc}(id) = \emptyset$
then return b' else return 0

Experiment $\mathbf{Exp}^{\text{hibe-ano-cpa-b}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
pick random oracle H
$(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{find}, pk)$
if $|id_0| \neq |id_1|$ or $|id_0| > d(k)$
or $|id_1| > d(k)$ or $M \notin \mathsf{MsgSp}(k)$
then return 0
$C \xleftarrow{\$} \mathsf{Enc}(pk, id_b, M)$
$b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{guess}, C, state)$
if $IDSet \cap (\text{anc}(id_0) \cup \text{anc}(id_1)) = \emptyset$ and
$\text{diff}(id_0, id_1) \subseteq L(k)$
then return b' else return 0

where the oracle $\text{KeyDer}(\cdot)$ is defined as

$$\text{if } |id| > d(k) \text{ then return } \perp \text{ ; } IDSet \leftarrow IDSet \cup \{id\} \text{ ; return } \mathsf{KeyDer}(msk, id) \text{ .}$$

We define the advantage of $\mathcal{A}$ in the corresponding experiments as

$$\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0}[d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right]$$

$$\mathbf{Adv}^{\text{hibe-ano-cpa}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{hibe-ano-cpa-1}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ano-cpa-0}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = 1\right]$$

The scheme $\mathcal{HIBE}$ is said to be HIBE-IND-CPA$[d]$-*secure* (resp. HIBE-ANO-CPA$[L, d]$-*secure*) if the respective advantage function is negligible for all PTAs $\mathcal{A}$.

F.5.2 A sufficient condition for anonymity

We further extend Lemma F.4.3 to the hierarchical case. To this end, we introduce a new notion HIBE-ANO-RE-CPA$[L, d]$ as follows. Let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a HIBE scheme with message space MsgSp, let L be a set of levels, and let d be the maximum hierarchy depth. To an adversary $\mathcal{A}$ and a bit b, we associate the following experiment:

Experiment $\mathbf{Exp}^{\text{hibe-ano-re-b}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k)$
$IDSet \leftarrow \emptyset$; $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$
pick random oracle H
$(id_0, id_1, M, state) \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{find}, pk)$
if $|id_0| \neq |id_1|$ or $|id_0| > d(k)$ or $|id_1| > d(k)$
or $M \notin \mathsf{MsgSp}(k)$ then return 0
$R \xleftarrow{\$} \{0,1\}^{|M|}$; $C \xleftarrow{\$} \mathsf{Enc}^H(pk, id_b, R)$
$b' \xleftarrow{\$} \mathcal{A}^{\text{KeyDer}(\cdot),H}(\texttt{guess}, C, state)$
if $IDSet \cap (\{id_0, id_1\} \cup \text{anc}(id_0) \cup \text{anc}(id_1)) = \emptyset$
and $\text{diff}(id_0, id_1) \subseteq L(k)$ then return b' else return 0

Oracle $\text{KeyDer}(id)$
if $|id| > d(k)$ then return $\perp$
$IDSet \leftarrow IDSet \cup \{id\}$
return $\mathsf{KeyDer}^H(msk, id)$

The HIBE-ANO-RE-CPA$[L, d]$-*advantage* of an adversary $\mathcal{A}$ in violating the level-L anonymity of the scheme $\mathcal{HIBE}$ with depth $d(k)$ is defined as

$$\mathbf{Adv}^{\text{hibe-ano-re}[L,d]}_{\mathcal{HIBE},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\text{hibe-ano-re-1}[L,d]}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{hibe-ano-re-0}[L,d]}_{\mathcal{IBE},\mathcal{A}}(k) = 1\right] .$$

A scheme $\mathcal{HIBE}$ is said to be HIBE-ANO-RE-CPA$[L, d]$-*secure* if this advantage is a negligible function in k for all PTAs $\mathcal{A}$. The following lemma follows from a hybrid argument similar to that of Lemma F.4.3.

Lemma F.5.1 Let $\mathcal{HIBE}$ be a HIBE scheme that is HIBE-IND-CPA$[d]$ and HIBE-ANO-RE-CPA$[L, d]$-secure for some set of levels L and hierarchy depth d. Then $\mathcal{HIBE}$ is also HIBE-ANO-CPA$[L, d]$-secure.

$\mathsf{Setup}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$; $P \xleftarrow{\$} \mathbb{G}_1^*$
$s_0 \xleftarrow{\$} \mathbb{Z}_p^*$; $S_0 \leftarrow 0$; $Q_0 \leftarrow s_0 P$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
$msk \leftarrow (pk, (), S_0, s_0)$
return (pk, msk)

$\mathsf{KeyDer}^{H_{1,1},\ldots,H_{1,l}}(usk, id)$
parse id as $(id_1, \ldots, id_{l+1})$
parse usk as $(pk, id|_l, S_l, Q_1, \ldots, Q_{l-1}, s_l)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
$S_{l+1} \leftarrow S_l + s_l H_{1,l+1}(id_{l+1})$
$Q_l \leftarrow s_l P$; $s_{l+1} \xleftarrow{\$} \mathbb{Z}_p^*$
return $(pk, id, S_{l+1}, Q_1, \ldots, Q_l, s_{l+1})$

$\mathsf{Enc}^{H_{1,1},\ldots,H_{1,l},H_2}(pk, id, M)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
parse id as $(id_1, \ldots, id_l)$
$r \xleftarrow{\$} \mathbb{Z}_p^*$; $C_1 \leftarrow rP$
for $i = 2, \ldots, l$ do $C_i \leftarrow r H_{1,i}(id_i)$
$C_{l+1} \leftarrow M \oplus H_2(e(rH_{1,1}(id_1), Q_0))$
return $(C_1, \ldots, C_{l+1})$

$\mathsf{Dec}^{H_2}(usk, C)$
parse usk as $(pk, id, S_l, Q_1, \ldots, Q_{l-1}, s_l)$
parse id as $(id_1, \ldots, id_l)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, Q_0)$
parse C as $(C_1, \ldots, C_{l+1})$
$\kappa \leftarrow e(S_l, C_1) \cdot \prod_{i=2}^{l} e(Q_{i-1}, C_i)^{-1}$
return $C_{l+1} \oplus H_2(\kappa)$

Figure F.6: Algorithms of the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme. $\mathcal{G}$ is a pairing parameter generator and $H_{1,i}$: $\{0,1\}^* \to \mathbb{G}_1^*$ and H_2: $\mathbb{G}_2 \to \{0,1\}^k$ are random oracles.

F.5.3 Construction

The HIBE scheme of [HL02] appears to be anonymous, but supports only two levels of identities, and is only resistant against limited collusions at the second level, and hence is not usable for our constructions that follow. Since the HIBE of [GS02], here denoted $\mathcal{GS}$-$\mathcal{HIBE}$, is equivalent to the Boneh-Franklin IBE scheme [BF03] when restricted to the first level, and since the latter is provably anonymous as per Theorem F.4.4, one could hope that $\mathcal{GS}$-$\mathcal{HIBE}$ is level-1 anonymous, but this turns out not to be true, and the HIBE of [BBG05] is not level-1 anonymous either. To see why, consider the following. The $\mathcal{GS}$-$\mathcal{HIBE}$ encryption of a message M under identity $id = (id_1, \ldots, id_l)$ is a tuple

$$(rP,\ rH_1(id|_2),\ \ldots,\ rH_1(id|_l),\ H_2(e(rP, H_1(id_1))) \oplus m) \tag{F.7}$$

where H_1, H_2 are random oracles, P is a generator of a pairing group that is part of pk, and r is chosen at random from $\mathbb{Z}_p$ by the encryption algorithm. Anonymity is violated because an adversary can decide whether a given ciphertext (C_1, C_2, C_3) is intended for $id = (id_1, id_2)$ or $id' = (id_1', id_2)$ by checking whether $e(C_2, P)$ equals $e(C_1, H_1(id))$ or $e(C_1, H_1(id'))$.

The lack of anonymity in $\mathcal{GS}$-$\mathcal{HIBE}$ stems from the fact that the hashes in the first l components of the ciphertext depend on the first component of the recipient's identity. In Figure F.6, we present a modified $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme that uses a different random oracle $H_{1,l}$ for each level l, and that computes ciphertexts as

$$(rP,\ rH_{1,2}(id_2),\ \ldots,\ rH_{1,l}(id_l),\ H_2(e(rP, H_{1,1}(id_1))) \oplus m) .$$

The following implies in particular that $m\mathcal{GS}$-$\mathcal{HIBE}$ is the first full HIBE scheme providing anonymity at any level. The restriction on d is inherited from [GS02]. We note that, subsequently to our work, Boyen and Waters [BW06] proposed a HIBE scheme that is anonymous at all levels in the standard (i.e., non-random-oracle) model.

Theorem F.5.2 For any $d(k) = O(\log(k))$, the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme is HIBE-ANO-CPA$[1, d]$-secure and HIBE-IND-CPA$[d]$-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.

We split up the proof in the following two lemmas. The proof of the first is given in Appendix F.9, and recycles ideas from [GS02, BF03]. We use Lemma F.5.1 to prove the second lemma.

Lemma F.5.3 For any $d(k) = O(\log(k))$, the mGS-HIBE scheme is HIBE-IND-CPA$[d]$-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.

Lemma F.5.4 For any $d(k) = O(\log(k))$, the mGS-HIBE scheme is HIBE-ANO-CPA$[1, d]$-secure in the random oracle model assuming the BDH problem is hard relative to the generator $\mathcal{G}$.

Proof: Given Lemmas F.5.1 and F.5.3, it suffices to show that mGS-HIBE is HIBE-ANO-RE-CPA$[1, d]$-secure. In the challenge ciphertext $(C_1^*, \ldots, C_{l+1}^*)$, the first component C_1 is chosen uniformly at random from $\mathbb{G}_1^*$. Component C_i^* for $2 \leq i \leq l$ is uniquely defined by C_1^* and the i-th component of the identity, which is the same for both challenge identities since they can only differ at level 1. Finally, if the message M is chosen uniformly at random from $\{0,1\}^k$, then the last component C_{l+1}^* is also uniformly distributed over $\{0,1\}^k$, independent of $H_2(e(rH_{1,1}(id_1), Q_0))$. Hence, the challenge ciphertext is identically distributed in both worlds, and the advantage of any adversary is 0. ∎

F.6 Public-key encryption with temporary keyword search

In a PEKS scheme, once the gateway has the trapdoor for a certain keyword, it can test whether this keyword was present in past ciphertexts, and can test its presence in any future ciphertexts. It may be useful to limit the period in which the trapdoor can be used. Here we propose an extension of PEKS that allows this. We call it public-key encryption with temporary keyword search (PETKS) or temporarily searchable encryption for short. A trapdoor here is created for a time interval $[s, e]$ and will only allow the gateway to test whether ciphertexts created in this time interval contain the keyword.

F.6.1 Definitions

PETKS SCHEMES. *Public-key encryption with temporary keyword search* (PETKS) is a generalization of PEKS in which a trapdoor can be issued for any desired window of time rather than forever. Formally, the scheme $\mathcal{PETKS} = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ consists of four PTAs and a function $N\colon \mathbb{N} \to \mathbb{N}$. Via $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the receiver generates its public and secret key; via $C \xleftarrow{\$} \mathsf{PETKS}^H(pk, w, i)$ a sender encrypts a keyword w in time period $i \in [0, N(k) - 1]$ to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(sk, w, s, e)$ the receiver computes a trapdoor t_w for keyword w in period $[s, e]$ where $0 \leq s \leq e \leq N(k) - 1$, and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$ the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". Here H is a random oracle whose domain and/or range might depend on k and pk. We require that for all $k \in \mathbb{N}$, all s, e, i with $0 \leq s \leq i \leq e \leq N(k) - 1$, and all $w \in \{0,1\}^*$,

$$\Pr\left[\, \mathsf{Test}^H(\mathsf{Td}^H(sk, w, s, e), \mathsf{PETKS}^H(pk, w, i)) = 1 \,\right] \;=\; 1\,,$$

where the probability is taken over the choice of $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^k)$, the random choice of H, and the coins of all the algorithms in the expression above.

CONSISTENCY. Consistency for PETKS schemes requires that no user $\mathcal{U}$ can output keywords w, w' and time period indices $s, e, i \in [1, N(k) - 1]$ such that $w \neq w'$ or $i \notin [s, e]$, yet still an encryption of w for time period i tests positively under a trapdoor for keyword w' and time period $[s, e]$. We define the advantage $\mathbf{Adv}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{U}}(k)$ as the probability that $\mathcal{U}$ succeeds in doing so. Just like for standard PEKS schemes, we distinguish between perfect, statistical and computational consistency.

PRIVACY. Privacy for a PETKS scheme asks that an adversary be unable to distinguish between the encryption of two challenge keywords of its choice in a time period $i \in [0, N(k)-1]$ of its choice, even if it is allowed not only to obtain trapdoors for non-challenge keywords issued for any time interval, but also is allowed to obtain trapdoors for *any* keywords (even the challenge ones), issued for time intervals not containing i. The formal experiment and the definition of PETKS-IND-CPA-advantage and security are otherwise analogous to those of standard PEKS schemes, and hence are omitted here.

F.6.2 Constructions for PETKS schemes

CONSTRUCTIONS WITH LINEAR COMPLEXITY. PETKS is reminiscent of forward-security [BM99, CHK03], and, as in these works, there are straightforward solutions with keys or trapdoors of length linear in $N(k)$. One such solution is to use a standard PEKS scheme and generate a different key pair (pk_i, sk_i) for each time period $i \in [0, N(k)-1]$. Let $pk = (pk_0, \ldots, pk_{N(k)-1})$ be the PETKS public key and $sk = (sk_0, \ldots, sk_{N(k)-1})$ be the PETKS secret key. During time period i, the sender encrypts a keyword w by encrypting w under pk_i using the PEKS scheme. The trapdoor for a keyword w in the interval $[s, e]$ consists of all PEKS trapdoors for w of periods $s, \ldots, e$. A somewhat more efficient solution is to let the PETKS master key pair be a single key pair for the standard PEKS scheme, and append the time period to the keyword (making sure that the string is uniquely decodable, e.g. by using a special separator symbol) when encrypting or computing trapdoors. This scheme achieves short public and secret keys, but still has trapdoor length linear in $N(k)$, because the PETKS trapdoor still contains PEKS trapdoors for all time periods $s, \ldots, e$. Note that both these construction only work for polynomially bounded $N(k)$.

THE hibe-2-petks TRANSFORM. We now present a transformation hibe-2-petks of a HIBE scheme into a PETKS scheme that yields a PETKS scheme with complexity logarithmic in $N(k)$ for all parameters. The construction is very similar to the generic construction of forward-secure encryption from binary-tree encryption [CHK03]. The number of time periods is $N(k) = 2^{t(k)}$ for some polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$. If $i \in [0, N(k)-1]$, then let $i_1 \ldots i_{t(k)}$ denote its binary representation as a $t(k)$-bit string. Intuitively, our construction instantiates a HIBE of depth $t(k)+1$ with keywords as the first level of the identity tree and the time structure on the lower levels. The trapdoor for keyword w and interval of time periods $[s, e]$ consists of the user secret keys of all identities from $(w, s_1, \ldots, s_{t(k)})$ to $(w, e_1, \ldots, e_{t(k)})$, but taking advantage of the hierarchical structure to include entire subtrees of keys.

More precisely, let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a HIBE scheme. Then we associate to it a PETKS scheme $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k)) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ such that $N(k) = 2^{t(k)}$, $\mathsf{KG}(1^k) = \mathsf{Setup}(1^k)$ and $\mathsf{PETKS}(pk, w, i) = (i, R, C)$ where $R \xleftarrow{\$} \{0,1\}^k$ and $C \leftarrow \mathsf{Enc}(pk, (w, i_1, \ldots, i_{t(k)}), R)$. The trapdoor algorithm $\mathsf{Td}(sk, w, s, e)$ first constructs a set T of identities as follows. Let j be the smallest index so that $s_j \neq e_j$. Then T is the set containing $(w, s_1, \ldots, s_{t(k)})$, $(w, e_1, \ldots, e_{t(k)})$, the right siblings of all nodes on the path from $(w, s_1, \ldots, s_{j+1})$ to $(w, s_1, \ldots, s_{t(k)})$, and the left siblings of all nodes on the path from $(w, e_1, \ldots, e_{j+1})$ to $(w, e_1, \ldots, e_{t(k)})$. If j does not exist, meaning $s = e$, then $T \leftarrow \{(w, s_1, \ldots, s_{t(k)})\}$. The trapdoor t_w is the set of tuples $((w, i_1, \ldots, i_r), \mathsf{KeyDer}(sk, (w, i_1, \ldots, i_r)))$ for all $(i_1, \ldots, i_r) \in T$. To test a ciphertext (i, R, C), the Test algorithm looks up a tuple $((w, i_1, \ldots, i_r), usk[(w, i_1, \ldots, i_r)])$ in t_w. It returns 0 when no such tuple is found. Otherwise, it derives $usk[(w, i_1, \ldots, i_{t(k)})]$ using repetitive calls to the KeyDer algorithm, and returns 1 iff $\mathsf{Dec}(usk[(w, i_1, \ldots, i_{t(k)})], C) = R$.

Theorem F.6.1 Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k))$ for some polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$. If $\mathcal{HIBE}$ is HIBE-ANO-CPA$[1, t(k)+1]$-secure,

then $\mathcal{PETKS}$ is PETKS-IND-CPA-secure. Furthermore, if $\mathcal{HIBE}$ is HIBE-IND-CPA$[t(k)+1]$-secure, then $\mathcal{PETKS}$ is computationally consistent.

We split the proof of the theorem over the following two lemmas.

Lemma F.6.2 Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k))$ for some polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$. If $\mathcal{HIBE}$ is HIBE-ANO-CPA$[1, t(k)+1]$-secure, then $\mathcal{PETKS}$ is PETKS-IND-CPA-secure.

Proof: Let $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ be a level-1 anonymous (HIBE-ANO-CPA$[1, t(k)+1]$-secure) HIBE scheme, and let $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k)) = (\mathsf{KG}, \mathsf{Td}, \mathsf{PETKS}, \mathsf{Test}, N)$ be the associated PETKS scheme. Given an adversary $\mathcal{A}$ breaking the PETKS-IND-CPA security of $\mathcal{PETKS}$, we construct an adversary $\mathcal{B}$ breaking the HIBE-ANO-CPA$[1, t(k)+1]$ security of $\mathcal{HIBE}$ as follows. On input public parameters pk, $\mathcal{B}$ runs $\mathcal{A}$ on inputs $(\texttt{find}, pk)$. When $\mathcal{A}$ queries its TRAPD oracle for the trapdoor of keyword w for interval $[s, e]$, then $\mathcal{B}$ constructs a set T exactly as the Td algorithm does, and constructs the corresponding trapdoor by querying its KEYDER oracle for the user secret keys corresponding to all identities in T.

When $\mathcal{A}$ outputs challenge keywords w_0, w_1 and time period i, $\mathcal{B}$ outputs challenge identities $id_0 = (w_0, i_1, \ldots, i_{t(k)})$, $id_1 = (w_1, i_1, \ldots, i_{t(k)})$ and a randomly chosen message M of length k. Note that identities id_0 and id_1 differ on level 1, but are otherwise equal, as required for level-1 anonymity. Upon receiving challenge ciphertext C, adversary $\mathcal{B}$ sends (i, R, C) to $\mathcal{A}$ and runs it until $\mathcal{A}$ outputs a bit b' (responding to $\mathcal{A}$'s oracle queries the same way as before). Adversary $\mathcal{B}$ outputs the same bit b'.

It is easy to see that, due to the ordered structure of the time tree, adversary $\mathcal{B}$ does not need to corrupt any ancestors of its challenge identities. Therefore, adversary $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does, and we have

$$\mathbf{Adv}^{\text{petks-ind-cpa}}_{\mathcal{PETKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ano-cpa}[1,t(k)+1]}_{\mathcal{HIBE},\mathcal{B}}(k)$$

for all $k \in \mathbb{N}$, from which the lemma follows. ∎

Lemma F.6.3 Let $\mathcal{HIBE}$ be a HIBE scheme, and let $\mathcal{PETKS} = \mathsf{hibe\text{-}2\text{-}petks}(\mathcal{HIBE}, t(k))$ for some polynomially bounded function $t : \mathbb{N} \to \mathbb{N}$. If $\mathcal{HIBE}$ is HIBE-IND-CPA$[t(k)+1]$-secure, then $\mathcal{PETKS}$ is computationally consistent.

Proof: Let $\mathcal{A}$ be an adversary of the consistency of $\mathcal{PETKS}$. We construct an HIBE-IND-CPA adversary $\mathcal{B}$ of $\mathcal{HIBE}$ as follows.

Adversary $\mathcal{B}^{\text{KEYDER}(\cdot)}(\texttt{find}, pk)$
  $(w, w', s, e, i) \xleftarrow{\$} \mathcal{A}(pk)$
  $R, R' \xleftarrow{\$} \{0,1\}^k$ (where $\{0,1\}^k$ is the message space of $\mathcal{HIBE}$)
  $id \leftarrow (w, i_1, \ldots, i_{t(k)})$
  $M_0 \leftarrow R$; $M_1 = R'$
  $state \leftarrow (pk, w, w', R, R', s, e, i)$
  return $(id, M_0, M_1, state)$

Adversary $\mathcal{B}^{\text{KEYDER}(\cdot)}(\texttt{guess}, C, state)$
  parse C as (i, R, C')
  $t_{w'} \xleftarrow{\$} \text{KEYDER}((w', i_1, \ldots, i_{t(k)}))$; $X \leftarrow \mathsf{Dec}(t_{w'}, C')$
  if $X = R'$ then return 1 else return 0

Since, by construction, $\mathsf{Test}(t_w, C)$ returns 0 whenever $i \notin [s, e]$, we can assume that $w' \neq w$ and $i \in [s, e]$. Then, exactly as in Theorem F.4.2, we have

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1}[t(k)+1]}_{\mathcal{HIBE},\mathcal{B}}(k) = 1\right] \geq \Pr\left[\mathbf{Exp}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{A}}(k) = 1\right] \quad \text{(F.8)}$$

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0}[t(k)+1]}_{\mathcal{HIBE},\mathcal{B}}(k) = 1\right] \leq 2^{-l}. \quad \text{(F.9)}$$

Equation (F.8) and Equation (F.9) give us

$$\mathbf{Adv}^{\text{petks-consist}}_{\mathcal{PETKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ind-cpa}[t(k)+1]}_{\mathcal{HIBE},\mathcal{B}}(k) + 2^{-l}.$$

The result follows. ∎

Complexity. Since the $m\mathcal{GS}$-$\mathcal{HIBE}$ has user secret keys and ciphertexts of size linear in the depth of the tree, our resulting PETKS scheme has public and secret keys of size $O(1)$, ciphertexts of size $O(\log N(k))$ and trapdoors of size $O(\log^2 N(k))$. We note that in this case a user can decrypt ciphertexts intended for any of its descendants directly, without needing to derive the corresponding secret key first. This makes the call to the KeyDer algorithm in the Test algorithm superfluous, thereby improving the efficiency of Test. Note that since the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme is only secure for tree depths $d(k) = O(\log(k))$, the derived PETKS scheme is restricted to a polynomial number of time periods.

Unbounded time periods. Using the techniques of [MMM02], one can create a variant of our scheme with efficiency depending on the number of *elapsed* time periods, rather than the maximal number of time periods $N(k)$. This means that there is no efficiency penalty for overestimating $N(k)$, so that a sufficiently high value can be chosen when setting up the system. However, for security reasons the number of time periods remains limited to a maximum of $N(k) \leq 2^{d(k)-\lceil \log d(k) \rceil - 1}$ periods, where $d(k)$ is the maximum depth of the underlying HIBE scheme.

F.7 Identity-based encryption with keyword search

In this section, we show how to combine the concepts of identity-based encryption and PEKS to obtain identity-based encryption with keyword search (IBEKS) or ID-based searchable encryption for short. Like in IBE schemes, this allows to use any string as a recipient's public key for the PEKS scheme.

F.7.1 Definitions

IBEKS schemes. An identity-based encryption with keyword search scheme $\mathcal{IBEKS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Td}, \mathsf{IBEKS}, \mathsf{Test})$ is made up of five algorithms. Via $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, where $k \in \mathbb{N}$ is the security parameter, the master generates the master keys; via $usk[id] \xleftarrow{\$} \mathsf{KeyDer}^H(msk, id)$, the master computes the secret key for identity id; via $C \xleftarrow{\$} \mathsf{IBEKS}^H(pk, id, w)$, a sender encrypts a keyword w to identity id to get a ciphertext; via $t_w \xleftarrow{\$} \mathsf{Td}^H(usk[id], w)$, the receiver computes a trapdoor t_w for keyword w and identity id and provides it to the gateway; via $b \leftarrow \mathsf{Test}^H(t_w, C)$, the gateway tests whether C encrypts w, where b is a bit with 1 meaning "accept" or "yes" and 0 meaning "reject" or "no". As usual H is a random oracle whose domain and/or range might depend on k and pk. For correctness, we require that for all $k \in \mathbb{N}$, all identities id, and all $w \in \{0,1\}^*$,

$$\Pr\left[\mathsf{Test}^H(\mathsf{Td}^H(\mathsf{KeyDer}^H(msk, id), w), \mathsf{IBEKS}^H(pk, id, w)) = 1\right] = 1\,,$$

where the probability is taken over the choice of $(pk, msk) \xleftarrow{\$} \mathsf{Setup}(1^k)$, the random choice of H, and the coins of all algorithms in the expression above.

CONSISTENCY. The notion of consistency for IBEKS is similar to the one given for PEKS. The advantage of a user $\mathcal{U}$ is defined as the probability that, on input the master public key pk, it can output keywords w, w' and identities id, id' such that $w \neq w'$ or $id \neq id'$, yet still an encryption of w under identity id tests positively under a trapdoor derived for keyword w' and identity id'. We again distinguish between perfect, statistical and computational consistency. Note that this definition also considers it a consistency problem if a trapdoor for identity id' tests positively for a ciphertext intended for identity $id \neq id'$. This type of problems is easily avoided by having the KeyDer, Td and IBEKS algorithms include the intended identity into the user secret keys, trapdoors and ciphertexts, respectively.

PRIVACY. We define privacy for IBEKS schemes says that an adversary should not be able to distinguish between the encryption of two different challenge keywords w_0, w_1 of its choice for any identity id of its choice. Moreover, this should be the case even if the adversary is allowed to obtain trapdoors for non-challenge keywords issued for any identity and to obtain trapdoors for w_0, w_1 for identities other than id. The advantage function $\mathbf{Adv}^{\text{ibeks-ind-cpa}}_{\mathcal{IBEKS},\mathcal{A}}(k)$ of an adversary $\mathcal{A}$ and the notion of IBEKS-IND-CPA security are defined analogously to standard PEKS schemes.

F.7.2 A generic transformation from anonymous HIBE schemes

We now propose a generic transform, called $\mathsf{hibe\text{-}2\text{-}ibeks}$, to convert any HIBE scheme with two levels into an IBEKS scheme. To obtain an IBEKS that is IBEKS-IND-CPA-secure, it is sufficient to start with a HIBE that is anonymous at level 2. Moreover, if the underlying HIBE is HIBE-IND-CPA[2]-secure, then the resulting IBEKS is also computationally consistent.

THE $\mathsf{hibe\text{-}2\text{-}ibeks}$ TRANSFORM. Given a HIBE scheme $\mathcal{HIBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ with two levels, $\mathsf{hibe\text{-}2\text{-}ibeks}$ returns the IBEKS scheme $\mathcal{IBEKS} = (\mathsf{Setup}, \overline{\mathsf{KeyDer}}, \mathsf{IBEKS}, \mathsf{Td}, \mathsf{Test})$ such that $\overline{\mathsf{KeyDer}}(msk, id) = (usk, id)$ where $usk \xleftarrow{\$} \mathsf{KeyDer}(msk, id)$, $\mathsf{IBEKS}(pk, id, w) = (id, R, C)$ where $R \xleftarrow{\$} \{0,1\}^k$ and $C = \mathsf{Enc}(pk, (id, w), R)$, $\mathsf{Td}(\overline{usk} = (usk, id), w) = (id, t_w)$ where $t_w \xleftarrow{\$} \mathsf{KeyDer}(usk, (id, w))$ and $\mathsf{Test}(\overline{t_w} = (id, t_w), (id', R, C))$ returns 1 iff $\mathsf{Dec}(t_w, C) = R$ and $id = id'$.

Theorem F.7.1 Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathcal{IBEKS} = \mathsf{hibe\text{-}2\text{-}ibeks}(\mathcal{HIBE})$. If $\mathcal{HIBE}$ is HIBE-IND-CPA[2]-secure, then $\mathcal{IBEKS}$ is computationally consistent. Furthermore, if $\mathcal{HIBE}$ is HIBE-ANO-CPA[2, 2]-secure, then $\mathcal{IBEKS}$ is IBEKS-IND-CPA-secure.

The proof of Theorem F.7.1 follows from Lemma F.7.2 and Lemma F.7.3.

Lemma F.7.2 Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathcal{IBEKS} = \mathsf{hibe\text{-}2\text{-}ibeks}(\mathcal{HIBE})$. If $\mathcal{HIBE}$ is HIBE-ANO-CPA[2, 2]-secure, then $\mathcal{IBEKS}$ is IBEKS-IND-CPA-secure.

Proof: Given an adversary $\mathcal{A}$ breaking the IBEKS-IND-CPA-security of $\mathcal{IBEKS}$, we construct an HIBE-ANO-CPA[2, 2]-adversary $\mathcal{B}$ breaking $\mathcal{HIBE}$ as follows. On input a public key pk, algorithm $\mathcal{B}$ runs $\mathcal{A}$ on the same input, answering $\mathcal{A}$'s KEYDER$(\cdot)$ queries by forwarding the output of its own KEYDER$(\cdot)$ oracle, and answering $\mathcal{A}$'s TRAPD(id, w) oracle queries by querying its own KEYDER$(\cdot)$ oracle for the secret key corresponding to identity (id, w). When $\mathcal{A}$ outputs a challenge identity $id^\star$ and two challenge keywords $w_0^\star, w_1^\star$, adversary $\mathcal{B}$ chooses a random message $M^\star \in \{0,1\}^k$ and outputs $M^\star$ as the challenge message and $id_0^\star = (id^\star, w_0^\star)$ and $id_1^\star = (id^\star, w_1^\star)$ as the challenge identities, which in fact differ only in the second entry. Let $C^\star$ be the challenge ciphertext that $\mathcal{B}$ receives at the beginning of its `guess` phase. Adversary $\mathcal{B}$ returns $(M^\star, C^\star)$ to

$\mathcal{A}$, and continues to run $\mathcal{A}$ (answering TRAPD queries the same way as before) until it outputs a bit b'. Algorithm $\mathcal{B}$ then outputs the same bit b' as its own output.

It is clear from the construction that $\mathcal{B}$'s simulation of $\mathcal{A}$'s environment is perfect. Since $\mathcal{A}$ cannot query its TRAPD oracle on keywords $(id^\star, w_0^\star)$ and $(id^\star, w_1^\star)$, $\mathcal{B}$ will not be forced to query its KEYDER on identities $id_0^\star$ and $id_1^\star$, and hence wins the game whenever $\mathcal{A}$ does. Therefore, we have that

$$\mathbf{Adv}^{\text{ibeks-ind-cpa}}_{\mathit{IBEKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\text{hibe-ano-cpa[2,2]}}_{\mathcal{HIBE},\mathcal{B}}(k) \ ,$$

from which the theorem follows for CPA security. This proves the lemma. ∎

Lemma F.7.3 Let $\mathcal{HIBE}$ be a HIBE scheme and let $\mathit{IBEKS} = \mathsf{hibe\text{-}2\text{-}ibeks}(\mathcal{HIBE})$. If $\mathcal{HIBE}$ is HIBE-IND-CPA[2]-secure, then IBEKS is computationally consistent.

Proof: Let $\mathcal{A}_1$ be an adversary of the consistency of IBEKS. We construct an HIBE-IND-CPA[2] adversary $\mathcal{B}_1$ of $\mathcal{HIBE}$ as follows.

Adversary $\mathcal{B}_1^{\text{KEYDER}(\cdot)}(\text{find}, pk)$
$(w, w', id, id') \xleftarrow{\$} \mathcal{A}_1(pk)$; $R, R' \xleftarrow{\$} \{0,1\}^l$ (where $\{0,1\}^l$ is the message space of $\mathcal{HIBE}$)
$id = (id', w)$
$w_0 = R$; $w_1 = R'$
$state = (pk, w, w', R, R')$
return $(id, w_0, w_1, state)$

Adversary $\mathcal{B}_1^{\text{KEYDER}(\cdot)}(\text{guess}, C, state)$
$t_{w'} \xleftarrow{\$} \text{KEYDER}((id', w'))$; $X \leftarrow \mathsf{Dec}(t_{w'}, C)$
if $X = R'$ then return 1 else return 0

Since, by construction, $\mathsf{Test}(t_w, C)$ returns 0 whenever $id \neq id'$, we can assume that $w' \neq w$ and $id' = id$. Thus, exactly as in Theorem F.4.2, we have

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-1[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) = 1\right] \geq \Pr\left[\mathbf{Exp}^{\text{ibeks-consist}}_{\mathit{IBEKS},\mathcal{A}}(k) = 1\right] \quad \text{(F.10)}$$

$$\Pr\left[\mathbf{Exp}^{\text{hibe-ind-cpa-0[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) = 1\right] \leq 2^{-l}. \quad \text{(F.11)}$$

Equation (F.10) and Equation (F.11) give us

$$\mathbf{Adv}^{\text{ibeks-consist}}_{\mathit{IBEKS},\mathcal{A}_1}(k) \leq \mathbf{Adv}^{\text{hibe-ind-cpa[2]}}_{\mathcal{HIBE},\mathcal{B}_1}(k) + 2^{-l}.$$

The result follows. ∎

F.7.3 Concrete instantiations

Neither the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme of [GS02] nor the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme of Figure F.6 are anonymous at the second level. For the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme, consider an adversary $\mathcal{A}$ who outputs challenge identities $id = (id_1, id_2)$ and $id' = (id_1, id_2')$ for any $id_1, id_2, id_2' \in \{0,1\}^*$ such that $id_2 \neq id_2'$, and any challenge message $M \in \{0,1\}^k$. When given the challenge ciphertext $C = (C_1, C_2, C_3)$, $\mathcal{A}$ checks whether $e(C_1, H_1(id)) = e(P, C_2)$. (See Equation (F.7) for how ciphertexts are created in the $\mathcal{GS}$-$\mathcal{HIBE}$ scheme.) If the test succeeds, then $\mathcal{A}$ returns 0, otherwise it returns 1. It is easy to see that the advantage of $\mathcal{A}$ is $\mathbf{Adv}^{\text{hibe-ano-cpa[2,2]}}_{\mathcal{GS}\text{-}\mathcal{HIBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$. A similar attack can be mounted on the $m\mathcal{GS}$-$\mathcal{HIBE}$ scheme by checking whether $e(C_1, H_{1,2}(id_2)) = e(P, C_2)$.

In Appendix F.8.3, we show that the recently introduced HIBE scheme by Boneh et al. [BBG05] is not level-2 anonymous either (and actually, not anonymous at any level). Subsequent to our work, Boyen and Waters [BW06] proposed a fully anonymous HIBE scheme that, when used to instantiate our generic construction, immediately yields an IBEKS scheme with security and consistency in the standard model.

F.7.4 Identity-based encryption with temporary keyword search

The ideas of Sections F.6 and F.7 can be further combined to create an identity-based encryption scheme with temporary keyword search. This can be constructed from a level-2 anonymous HIBE scheme by putting the users' identities at the first level of the hierarchy, the keywords at the second, and a binary tree of time frames on the levels below.

Acknowledgements

We thank Nigel Smart for suggesting the concept of temporarily searchable encryption. Second and tenth authors were supported in part by NSF grants ANR-0129617 and CCR-0208842 and by an IBM Faculty Partnership Development Award. The fourth author was supported by the research program Sentinels (`http://www.sentinels.nl`). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. Fifth author was supported by an IBM Ph.D Fellowship. Eighth author is a Postdoctoral Fellow of the Research Foundation – Flanders (FWO), and was supported in part by the Flemish Government under GOA Mefisto 2006/06 and Ambiorix 2005/11, and by the European Commission through the IST Project PRIME. The rest of the authors were supported in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

F.8 Appendix: Attacks against the anonymity of existing schemes

F.8.1 Waters' IBE scheme

We recall Waters' IBE scheme [Wat05] $\mathcal{W}\text{-}\mathcal{IBE} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Enc}, \mathsf{Dec})$ in Figure F.7. Associated with $\mathcal{W}\text{-}\mathcal{IBE}$ is a polynomial n. It is assumed that all user identities are $n(k)$-bit (e.g. 160-bit) strings (for instance obtained by hashing the actual identity using a collision-resistant hash function), which are written as $id = id[1]id[2]\ldots id[n]$, where each $id[i]$ $(1 \leq i \leq n)$ is a bit $id[i] \in \{0,1\}$. (We drop the argument k to n when k is understood.) The message space is defined by $\mathsf{MsgSp}(k) = \{0,1\}^k$, and messages are encoded as elements of $\mathbb{G}_2$ in the scheme.

We now describe a PTA $\mathcal{A}$ against the IBE-ANO-CPA-security of $\mathcal{W}\text{-}\mathcal{IBE}$. In the `find` stage it gets input a public key $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E)$, and returns any two distinct n-bit strings id_0, id_1 as challenge identities, along with any k-bit challenge message. In the `guess` phase, given a challenge ciphertext $C = (C_1, C_2, C_3)$ formed by encrypting M under id_b, where $b \in \{0,1\}$ is the challenge bit, it computes $V' \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id_1[i]\boldsymbol{U}[i]$. If $e(P, C_3) = e(C_2, V')$ then it returns 1 else it returns 0. It is easy to see that $\mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{W}\text{-}\mathcal{IBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

F.8.2 Boneh-Boyen's IBE scheme

The IBE scheme by Boneh and Boyen [BB04b], here referred to as $\mathcal{BB}\text{-}\mathcal{IBE}$, is depicted in Figure F.8. An identity is represented by a vector of $n(k)$ symbols $id[1\ldots n] \in \Sigma^n$ where Σ is an alphabet of size s. In the original scheme, these are obtained as the output of an admissible hash function, but we ignore this here as it is irrelevant to the attack.

$\mathsf{Setup}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
$P, Q \xleftarrow{\$} \mathbb{G}_1^*$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$; $P_1 \leftarrow \alpha P$; $Q_1 \leftarrow \alpha Q$
$\boldsymbol{U}[0 \ldots n] \xleftarrow{\$} \mathbb{G}_1^{n+1}$; $E \leftarrow e(P, Q)$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E)$; $msk \leftarrow (pk, Q_1)$
return (pk, msk)

$\mathsf{KeyDer}(msk, id)$
parse msk as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E), Q_1)$
$r \xleftarrow{\$} \mathbb{Z}_p$; $V \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id[i]\boldsymbol{U}[i]$
$usk[id] \leftarrow (Q_1 + rV, rP)$
return $usk[id]$

$\mathsf{Enc}(pk, id, M)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, E)$
$V \leftarrow \boldsymbol{U}[0] + \sum_{i=1}^{n} id[i]\boldsymbol{U}[i]$
$t \xleftarrow{\$} \mathbb{Z}_p$; $T \leftarrow E^t$
$C \leftarrow (T \cdot M, tP, tV)$
return C

$\mathsf{Dec}(usk[id], C)$
parse $usk[id]$ as (S_1, S_2), C as (C_1, C_2, C_3)
$T' \leftarrow e(S_1, C_2) \cdot e(S_2, C_3)^{-1}$
return $T'^{-1} \cdot C_1$

Figure F.7: The algorithms constituting $\mathcal{W}$-$\mathcal{IBE}$. Identities are represented as bit strings $id = id[1, \ldots, n] \in \{0,1\}^n$.

$\mathsf{Setup}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
$P, Q \xleftarrow{\$} \mathbb{G}_1^*$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$; $P_1 \leftarrow \alpha P$; $Q_1 \leftarrow \alpha Q$
$\boldsymbol{U}[1 \ldots n, 1 \ldots s]) \xleftarrow{\$} \mathbb{G}_1^{n \times s}$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, \boldsymbol{U})$; $msk \leftarrow (pk, Q_1)$
return (pk, msk)

$\mathsf{KeyDer}(msk, id)$
parse msk as $((\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, Q), Q_1)$
$r_1, \ldots, r_n \xleftarrow{\$} \mathbb{Z}_p$; $V \leftarrow \sum_{i=1}^{n} r_i \boldsymbol{U}[i, id[i]]$
$usk[id] \leftarrow (Q_1 + V,\ r_1 P,\ \ldots,\ r_n P)$
return $usk[id]$

$\mathsf{Enc}(pk, id, M)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, \boldsymbol{U}, Q)$
$t \xleftarrow{\$} \mathbb{Z}_p$; $T \leftarrow e(P_1, Q)^t$
$C \leftarrow (T \cdot M, tP, t\boldsymbol{U}[1, id[1]], \ldots, t\boldsymbol{U}[n, id[n]])$
return C

$\mathsf{Dec}(usk[id], C)$
parse $usk[id]$ as $(S_1, S_2, \ldots, S_{n+1})$
parse C as $(C_1, \ldots, C_{n+2})$
$T' \leftarrow e(S_1, C_2) \cdot \prod_{i=1}^{n} e(S_{i+1}, C_{i+2})^{-1}$
return $T'^{-1} \cdot C_1$

Figure F.8: The algorithms constituting $\mathcal{BB}$-$\mathcal{IBE}$. Identities are represented as vectors of symbols $id = id[1, \ldots, n] \in \Sigma^n$, where $|\Sigma| = s$.

Consider a PTA $\mathcal{A}$ that, on input $pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, \boldsymbol{U})$, outputs any two distinct identities $id_0, id_1 \in \Sigma^n$, and any message $M \in \{0,1\}^k$. Let $i \in \{1, \ldots, n\}$ be an index so that $id_0[i] \neq id_1[i]$. When $\mathcal{A}$ is given the challenge ciphertext $C = (C_1, \ldots, C_{n+2})$, it checks whether $e(C_2, \boldsymbol{U}[i, id_0[i]]) = e(P, C_{i+2})$. If so, then $\mathcal{A}$ returns 0, else it returns 1. It is easily verified that $\mathbf{Adv}^{\text{ibe-ano-cpa}}_{\mathcal{BB}\text{-}\mathcal{IBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

F.8.3 Boneh-Boyen-Goh's HIBE scheme

The recently proposed $\mathcal{BBG}$-$\mathcal{HIBE}$ scheme [BBG05], depicted in Figure F.9, is not anonymous at any single level, and therefore not at any set of multiple levels either. This can be seen from the following adversary $\mathcal{A}$ that breaks the anonymity at level l. On input $pk = (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$, adversary $\mathcal{A}$ outputs challenge identities $(id_1, \ldots, id_{l-1}, id_l)$ and $(id_1, \ldots, id_{l-1}, id'_l)$ for any $id_1, \ldots, id_l, id'_l \in \mathbb{Z}_p$ such that $id_l \neq id'_l$, and any challenge message $M \in \{0,1\}^k$. When given the challenge ciphertext $C = (C_1, C_2, C_3)$, $\mathcal{A}$ checks whether $e(C_2, id_1\boldsymbol{U}[1] + \ldots + id_l\boldsymbol{U}[l] + Q_2) = e(P, C_3)$. If this is the case, then $\mathcal{A}$ returns 0, otherwise it returns 1. It is easily verified that $\mathbf{Adv}^{\text{hibe-ano-cpa}[l,d]}_{\mathcal{BBG}\text{-}\mathcal{HIBE},\mathcal{A}}(k) \geq 1 - 2^{-k}$.

$\mathsf{Setup}(1^k)$
$(\mathbb{G}_1, \mathbb{G}_2, p, e) \xleftarrow{\$} \mathcal{G}(1^k)$
$P \xleftarrow{\$} \mathbb{G}_1$; $\alpha \xleftarrow{\$} \mathbb{Z}_p$; $P_1 \leftarrow \alpha P$
$Q, Q_2 \xleftarrow{\$} \mathbb{G}_1$; $Q_1 \leftarrow \alpha Q$
$\boldsymbol{U}[1 \ldots d(k)]) \xleftarrow{\$} \mathbb{G}_1^{d(k)}$
$pk \leftarrow (\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
$msk \leftarrow (pk, Q_1, 0, \ldots, 0)$
return (pk, msk)

$\mathsf{KeyDer}(usk, id)$
$l \leftarrow |id|$; parse usk as $(pk, A, B, S_l, \ldots, S_{d(k)})$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
parse id as $(id_1, \ldots, id_l)$; $r \xleftarrow{\$} \mathbb{Z}_p$
$A' \leftarrow A + id_l S_l + r(id_1 \boldsymbol{U}[1] + \ldots + id_l \boldsymbol{U}[l] + Q_2)$
$B' \leftarrow B + rP$
for $l+1 \le i \le d(k)$ do $S'_i \leftarrow S_i + r\boldsymbol{U}[i]$
return $(pk, A', B', S'_{l+1}, \ldots, S'_{d(k)})$

$\mathsf{Enc}(pk, id, M)$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
parse id as $(id_1, \ldots, id_l)$
$t \xleftarrow{\$} \mathbb{Z}_p$; $T \leftarrow e(P_1, Q)^t$
$C_3 \leftarrow t(id_1 \boldsymbol{U}[1] + \ldots + id_l \boldsymbol{U}[l] + Q_2)$
$C \leftarrow (T \cdot M, tP, C_3)$
return C

$\mathsf{Dec}(usk, C)$
parse usk as $(pk, A, B, S_l, \ldots, S_{d(k)})$
parse pk as $(\mathbb{G}_1, \mathbb{G}_2, p, e, P, P_1, Q, Q_2, \boldsymbol{U})$
parse C as (C_1, C_2, C_3)
$T' \leftarrow e(A, C_2) \cdot e(B, C_3)^{-1}$
return $T'^{-1} \cdot C_1$

Figure F.9: The algorithms constituting $\mathcal{BBG}$-$\mathcal{HIBE}$ with maximum hierarchy depth $d(k)$. An identity at level l is represented as a vector $id = (id_1, \ldots, id_l) \in \mathbb{Z}_p^l$.

F.9 Appendix: Proof of Lemma F.5.3

Suppose that there is an adversary $\mathcal{A}$ of $\mathcal{MGS}$-$\mathcal{HIBE}$ that breaks its HIBE-IND-CPA$[d]$ security. We will show how to use $\mathcal{A}$ in the construction of a simulator $\mathcal{B}$ that solves the bilinear Diffie-Hellman problem. Let $n_{1,i}$ be the number of queries that $\mathcal{A}$ makes to the $H_{1,i}$ oracle, let n_2 be the number of queries to the H_2 oracle, let n_e be the number of queries to the key extraction oracle, and let $n_h = \sum_{i-1}^{d(k)} n_{1,i} + n_2$ be the total number of hash queries.

The simulator is given as input (P, aP, bP, cP). It sets $Q_0 \leftarrow bP$ as the public key and then runs $\mathcal{A}(\texttt{find}, pk)$. The simulator responds to $\mathcal{A}$'s queries as described below. To maintain consistency between queries it keeps lists $L_{1,1}, \ldots, L_{1,d(k)}$, L_2 and L_3. All lists are initially empty. At the very beginning the simulator chooses $n^*_{1,i} \xleftarrow{\$} \{1, \ldots, n_{1,i}\}$ and $s^*_i, x^*_i \xleftarrow{\$} \mathbb{Z}_p^*$, and it computes $Q^*_i \leftarrow s^*_i(bP)$ for $1 \le i \le d(k)$.

For the description of the simulation we distinguish between $H_{1,1}$ queries and $H_{1,i}$ queries for $i \ge 2$. Without loss of generality, we assume that before querying the KeyDer oracle to obtain the secret key of $id = (id_1, \ldots, id_l)$, adversary $\mathcal{A}$ first queried $H_{1,i}(id_i)$ for all $1 \le i \le l$.

$H_{1,1}$ **Queries:** To respond to a query id_1, proceed as follows.

- If $L_{1,1}$ contains $(id_1, P_1, *)$ for some P_1, respond with P_1.
- If this is the $n^*_{1,1}$-th call to the $H_{1,1}$ oracle, let $id^*_1 \leftarrow id_1$, add $(id^*_1, aP, \perp)$ to $L_{1,1}$ and respond with aP.
- Else, randomly choose an integer $x_1 \xleftarrow{\$} \mathbb{Z}_p^*$, add (id_1, x_1P, x_1) to $L_{1,1}$ and reply with x_1P.

$H_{1,i}$ **Queries,** $i \ge 2$**:** To respond to a query id_i, proceed as follows.

- If $L_{1,i}$ contains $(id_i, P_i, *)$ for some P_i then respond with P_i.
- If this is the $n^*_{1,i}$-th query to the $H_{1,i}$ oracle, let $id^*_i \leftarrow id_i$, add (id^*_i, x^*_iP, x^*_i) to $L_{1,i}$ and respond with x^*_iP.

- Else, choose an integer $x_i \xleftarrow{\$} \mathbb{Z}_p^*$ and compute $P_i \leftarrow x_i P - {s_{i-1}^*}^{-1}(aP + \sum_{j=2}^{i-1} s_{j-1}^* x_j^* P)$. If $P_i = 0$, then abort; else, add (id_i, P_i, x_i) to $L_{1,i}$ and reply with P_i.

H_2 **Queries:** To respond to a query κ, proceed as follows.

- If $(\kappa, K) \in L_2$ for some K, respond with K.
- Else, choose K uniformly at random from $\{0,1\}^n$, respond with K and add (κ, K) to L_2.

KeyDer **Queries:** To respond to a query $id = (id_1, \ldots, id_l)$, proceed as follows.

- If $(id_1, \ldots, id_l) = (id_1^*, \ldots, id_l^*)$, then $\mathcal{B}$ aborts.
- Let j be the largest integer $1 \leq j \leq l$ so that $(id|_j, S_j, Q_1, \ldots, Q_{j-1}, s_j) \in L_3$, or let $j = 0$ if such element does not exist.
- For $i = j+1, \ldots, l$, do the following:
  - Find $(id_i, P_i, x_i) \in L_{1,i}$.
  - If $i = 1$ and $id_1 = id_1^*$, then add $(id_1^*, \perp, \perp)$ to L_3. If $i = 1$ and $id_1 \neq id_1^*$, then compute $S_1 \leftarrow x_i(bP)$, choose $s_1 \xleftarrow{\$} \mathbb{Z}_p^*$, and add (id_1, S_1, s_1) to L_3.
  - If $i > 1$ and $S_{i-1} \neq \perp$, then look up (id_i, P_i, x_i) in $L_{1,i}$, compute $S_i \leftarrow S_{i-1} + s_{i-1}P_i$, $Q_{i-1} \leftarrow s_{i-1}P$, choose $s_i \xleftarrow{\$} \mathbb{Z}_p^*$, and add $(id|_i, S_i, Q_1, \ldots, Q_{i-1}, s_i)$ to L_3.
  - If $i > 1$ and $S_{i-1} = \perp$ and $id_i = id_i^*$, then compute $Q_{i-1} \leftarrow s_{i-1}^*(bP)$ and add $(id|_i, \perp, Q_1, \ldots, Q_{i-1}, \perp)$ to L_3.
  - If $i > 1$, $S_{i-1} = \perp$ and $id_i \neq id_i^*$, then look up (id_i, P_i, x_i) in $L_{1,i}$, compute $S_i \leftarrow s_{i-1}^* x_i P$, let $Q_{i-1} \leftarrow s_{i-1}^*(bP)$, choose $s_i \xleftarrow{\$} \mathbb{Z}_p^*$, and add $(id|_i, S_i, Q_1^*, \ldots, Q_{i-1}^*, s_i)$ to L_3.
- Find $(id, S_l, Q_1, \ldots, Q_{l-1}, s_l) \in L_3$ and return $(id, S_l, Q_1, \ldots, Q_{l-1}, s_l)$.

At some point $\mathcal{A}$ outputs $(id = (id_1, id_2, \ldots, id_l), M_0, M_1, state)$. Without loss of generality, we assume that the adversary submitted id_i to the $H_{1,i}$ oracle before for all $1 \leq i \leq l$. If $id \neq (id_1^*, \ldots, id_l^*)$, then $\mathcal{B}$ aborts. Otherwise, he sets $C_1^* \leftarrow cP$, $C_2^* \leftarrow x_2^*(cP)$, $\ldots$, $C_l \leftarrow x_l^*(cP)$, he chooses C_{l+1}^* uniformly at random from $\{0,1\}^n$, and lets $C^* \leftarrow (C_1^*, C_2^*, \ldots, C_{l+1}^*)$. He then proceeds to run $\mathcal{A}(\texttt{guess}, C^*, state)$. Once $\mathcal{A}$ completes its attack by outputting its guess b', the simulator chooses a random element (κ, K) from L_2 and outputs κ as its solution to the bilinear Diffie-Hellman problem.

We first show that our simulator $\mathcal{B}$ provides a real attack environment for $\mathcal{A}$ as long as $\mathcal{B}$ doesn't abort. The public key pk given to $\mathcal{A}$ is correctly distributed because the challenge elements aP, bP, cP are random elements from $\mathbb{G}_1^*$. The responses to $H_{1,i}$ queries are uniformly distributed over $\mathbb{G}_1^*$ due to the independent random choices of x_i (when simulating queries $H_{1,i}(id_i)$, $id_i \neq id_i^*$, $1 \leq i \leq d(k)$), of x_i^* (which is used to simulate $H_{1,i}(id_i^*)$ queries, $2 \leq i \leq d(k)$) and due to the uniform distribution of aP (which is used to simulate $H_{1,1}(id_1^*)$). Responses to H_2 queries are easily seen to be correctly distributed. The way KeyDer queries are handled requires a bit more explanation. For all level-1 identities $id_1 \neq id_1^*$, the returned secret key (S_1, s_1) contains the unique group element S_1 such that $e(Q_0, H_{1,1}(id_1)) = e(S_1, P)$ and a uniformly distributed scalar s_1, as in the real game. For all descendants of $id_1 \neq id_1^*$, the secret keys are derived from (S_1, s_1) exactly as in the real scheme. Now consider identity $(id_1^*, \ldots, id_{i-1}^*, id_i)$ with $id_i \neq id_i^*$, for which a tuple $(S_i, Q_1, \ldots, Q_{i-1}, s_i)$ is returned as the secret key. The values $Q_1, \ldots, Q_{i-2}$ are inherited from the ancestors, as in the real scheme; Q_{i-1} is a random group element due to the random choice of s_{i-1}^*; and s_i is a random element in $\mathbb{Z}_p^*$. The simulated value $S_i = s_{i-1}^* x_i(bP)$ is then the unique group element such that $e(H_{1,1}(id_1^*), Q_0) =$

$e(S_i, P) \cdot \prod_{j=2}^{i-1} e(H_{1,j}(id_j^*), Q_{j-1})^{-1} \cdot e(H_{1,i}(id_i), Q_{i-1})^{-1}$, as required by the scheme. This can be seen from:

$$\begin{aligned} & e(H_{1,1}(id_1^*),\, Q_0) \cdot \prod_{j=2}^{i-1} e(H_{1,j}(id_j^*),\, Q_{j-1}) \cdot e(H_{1,i}(id_i),\, Q_{i-1}) \\ = \;& e(aP,\, bP) \cdot \prod_{j=2}^{i-1} e(x_j^* P,\, s_{j-1}^* bP) \cdot e\big(x_i P - {s_{i-1}^*}^{-1}(aP + \textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P),\, s_{i-1}^* bP\big) \\ = \;& e(aP,\, bP) \cdot e\big(\textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P,\, bP\big) \cdot e\big(s_{i-1}^* x_i P - aP - \textstyle\sum_{j=2}^{i-1} s_{j-1}^* x_j^* P,\, bP\big) \\ = \;& e(S_3, P)\ . \end{aligned}$$

The secret keys of descendants of these nodes are derived from $(S_i, Q_1, \ldots, Q_{i-1}, s_i)$ as dictated by the scheme, and hence are correctly distributed as well.

The only part of $\mathcal{A}$'s environment left to analyze is the challenge ciphertext $C^* = (C_1^*, \ldots, C_{l+1}^*)$. The first component $C_1^* = cP$ is uniformly distributed over $\mathbb{G}_1^*$, and the second to l-th components are the unique group elements such that $e(C_i^*, P) = e(C_1^*, H_{1,i}(id_i^*))$ for $2 \le i \le l$. The last component C_{l+1}^* however may deviate from the distribution in a real game, depending on $\mathcal{A}$'s H_2 queries. In the following, we show that this does not harm our analysis, intuitively because the only way $\mathcal{A}$ can distinguish between the real and the simulated game is by making an H_2 query that helps $\mathcal{B}$ solve the BDH problem.

Let s_0 be the master secret key of the scheme in a real HIBE-IND-CPA[d] attack on $\mathit{mGS\text{-}HIBE}$, and let $D \leftarrow e(s_0 H_{1,1}(id_1),\ C_1^*)$. Let Ask be the event that $\mathcal{A}$ queries the H_2 oracle on point D. Let $\Pr_{\mathrm{R}}[\,\cdot\,]$ denote the probability of an event taking place in a real attack on $\mathit{mGS\text{-}HIBE}$, and let $\Pr_{\mathcal{B}}[\,\cdot\,]$ denote the probability in the environment simulated by $\mathcal{B}$. We argue that $\Pr_{\mathrm{R}}[\,\textsc{Ask}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$, as long as $\mathcal{B}$ doesn't abort. Let $\textsc{Ask}_i$ be the event that $\mathcal{A}$ queries $H_2(D)$ within the first i queries to H_2. Obviously, $\Pr_{\mathrm{R}}[\,\textsc{Ask}_0\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_0\,] = 0$. Now assume that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_{i-1}\,]$. We have that

$$\begin{aligned} \Pr_{\mathrm{R}}[\,\textsc{Ask}_i\,] &= \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \textsc{Ask}_{i-1}\,] \cdot \Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] \\ &\quad + \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,] \cdot \Pr_{\mathrm{R}}[\,\neg\textsc{Ask}_{i-1}\,] \\ &= \Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] + \Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,] \cdot \Pr_{\mathrm{R}}[\,\neg\textsc{Ask}_{i-1}\,]\ . \end{aligned}$$

We know that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_{i-1}\,]$, so we only have to show that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_i \mid \neg\textsc{Ask}_{i-1}\,]$. Given that $\neg\textsc{Ask}_{i-1}$ and that $\mathcal{B}$'s simulation didn't abort, the simulated public key, the oracle responses and the first l components of the ciphertext provided by $\mathcal{B}$ are distributed exactly as in a real attack, as we explained before. Moreover, since $\mathcal{A}$ did not query for $H_2(D)$ yet, from $\mathcal{A}$'s point of view the last ciphertext component C_{l+1}^* is a random string in $\{0,1\}^n$, both in the real attack and in the simulated environment. Since all the information on which $\mathcal{A}$ can base its decision for its next H_2 query is identically distributed in both environments, the probability that $\mathcal{A}$ chooses to query D is the same in both environments as well. Hence, we have that $\Pr_{\mathrm{R}}[\,\textsc{Ask}_i\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}_i\,]$, and by induction that $\Pr_{\mathrm{R}}[\,\textsc{Ask}\,] = \Pr_{\mathcal{B}}[\,\textsc{Ask}\,]$.

The probability that $\mathcal{A}$ wins a real attack against $\mathit{mGS\text{-}HIBE}$ can be written as

$$\begin{aligned} \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins}\,] &= \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \textsc{Ask}\,] + \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \neg\textsc{Ask}\,] \\ &= \Pr_{\mathrm{R}}[\,\mathcal{A}\ \textsc{wins} \wedge \textsc{Ask}\,] + \frac{1}{2} \\ &\le \Pr_{\mathrm{R}}[\,\textsc{Ask}\,] + \frac{1}{2}\ , \end{aligned}$$

where the second equation is true because in the event $\neg\text{ASK}$, the distribution of the challenge ciphertext is completely independent of M_0, M_1, and hence the probability that $\mathcal{A}$ guesses correctly is 1/2. Since $\Pr_{\text{R}}[\text{ASK}] = \Pr_{\mathcal{B}}[\text{ASK}]$ and moreover

$$\Pr_{\text{R}}[\mathcal{A} \text{ WINS}] = \frac{1}{2} \cdot \mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{m\mathcal{GS}\text{-}\mathcal{HIBE},\mathcal{A}}(k) + \frac{1}{2} ,$$

it follows that

$$\Pr_{\mathcal{B}}[\text{ASK}] \geq \frac{1}{2} \cdot \mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{m\mathcal{GS}\text{-}\mathcal{HIBE},\mathcal{A}}(k) .$$

Now we only have to relate $\mathcal{B}$'s advantage in solving the BDH problem to $\Pr_{\mathcal{B}}[\text{ASK}]$. In the game simulated by $\mathcal{B}$, the probability that $\mathcal{B}$ guesses the correct identities such that $id = (id_1^*, \ldots, id_l^*)$ $1/\prod_{i=1}^{l} n_{1,i} \geq n_h^{-d(k)}$; the probability that $\mathcal{B}$ guesses the correct H_2 query is $1/n_2 \geq n_h^{-1}$; and the probability that $\mathcal{B}$ aborts when answering $H_{1,i}$ queries is $\sum_{i=1}^{d(k)} n_{1,i}/(p-1) \leq n_h/2^k$. The advantage of $\mathcal{B}$ in solving the BDH problem is

$$\mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{B}}(k) \geq \frac{1}{n_h^{d(k)+1}} \cdot \left(1 - \frac{n_h}{2^k}\right) \cdot \Pr_{\mathcal{B}}[\text{ASK}]$$

and hence

$$\mathbf{Adv}^{\text{hibe-ind-cpa}[d]}_{m\mathcal{GS}\text{-}\mathcal{HIBE},\mathcal{A}}(k) \leq 2 \cdot n_h^{d(k)+1} \cdot \mathbf{Adv}^{\text{bdh}}_{\mathcal{G},\mathcal{B}}(k) + \frac{n_h}{2^k} ,$$

from which the theorem follows.

Appendix G

Robust Encryption

TCC 2010
[ABN10] with M. Bellare and G. Neven

Abstract : *We provide a provable-security treatment of "robust" encryption. Robustness means it is hard to produce a ciphertext that is valid for two different users. Robustness makes explicit a property that has been implicitly assumed in the past. We argue that it is an essential conjunct of anonymous encryption. We show that natural anonymity-preserving ways to achieve it, such as adding recipient identification information before encrypting, fail. We provide transforms that do achieve it, efficiently and provably. We assess the robustness of specific encryption schemes in the literature, providing simple patches for some that lack the property. We explain that robustness of the underlying anonymous IBE scheme is essential for PEKS (Public Key Encryption with Keyword Search) to be consistent (meaning, not have false positives), and our work provides the first generic conversions of anonymous IBE schemes to consistent (and secure) PEKS schemes. Overall our work enables safer and simpler use of encryption.*

G.1 Introduction

This paper provides a provable-security treatment of encryption "robustness." Robustness reflects the difficulty of producing a ciphertext valid under two different encryption keys. The value of robustness is conceptual, "naming" something that has been undefined yet at times implicitly (and incorrectly) assumed. Robustness helps make encryption more mis-use resistant. We provide formal definitions of several variants of the goal; consider and dismiss natural approaches to achieve it; provide two general robustness-adding transforms; test robustness of existing schemes and patch the ones that fail; and discuss some applications.

The definitions. Both the PKE and the IBE settings are of interest and the explication is simplified by unifying them as follows. Associate to each identity an *encryption key*, defined as the identity itself in the IBE case and its (honestly generated) public key in the PKE case. The adversary outputs a pair id_0, id_1 of distinct identities. For strong robustness it also outputs a ciphertext C^*; for weak, it outputs a message M^*, and C^* is defined as the encryption of M^* under the encryption key ek_1 of id_1. The adversary wins if the decryptions of C^* under the decryption keys dk_0, dk_1 corresponding to ek_0, ek_1 are *both* non-$\perp$. Both weak and strong robustness can be considered under chosen plaintext or chosen ciphertext attacks, resulting in

four notions (for each of PKE and IBE) that we denote WROB-CPA, WROB-CCA, SROB-CPA, SROB-CCA.

WHY ROBUSTNESS? The primary security requirement for encryption is data-privacy, as captured by notions IND-CPA or IND-CCA [GM84, RS92, DDN00, BDPR98, BF03]. Increasingly, we are also seeing a market for *anonymity*, as captured by notions ANO-CPA and ANO-CCA [BBDP01, ABC+08]. Anonymity asks that a ciphertext does not reveal the encryption key under which it was created.

Where you need anonymity, there is a good chance you need robustness too. Indeed, we would go so far as to say that robustness is an essential companion of anonymous encryption. The reason is that without it we would have security without basic communication correctness, likely upsetting our application. This is best illustrated by the following canonical application of anonymous encryption, but shows up also, in less direct but no less important ways, in other applications. A sender wants to send a message to a *particular* target recipient, but, to hide the identity of this target recipient, anonymously encrypts it under her key and broadcasts the ciphertext to a larger group. But as a member of this group I need, upon receiving a ciphertext, to know whether or not I am the target recipient. (The latter typically needs to act on the message.) Of course I can't tell whether the ciphertext is for me just by looking at it since the encryption is anonymous, but decryption should divulge this information. It does, unambiguously, if the encryption is robust (the ciphertext is for me iff my decryption of it is not $\perp$) but otherwise I might accept a ciphertext (and some resulting message) of which I am not the target, creating mis-communication. Natural "solutions," such as including the encryption key or identity of the target recipient in the plaintext before encryption and checking it upon decryption, are, in hindsight, just attempts to add robustness without violating anonymity and, as we will see, don't work.

We were lead to formulate robustness upon revisiting Public key Encryption with Keyword Search (PEKS) [BDOP04]. In a clever usage of anonymity, Boneh, Di Crescenzo, Ostrovsky and Persiano (BDOP) [BDOP04] showed how this property in an IBE scheme allowed it to be turned into a privacy-respecting communications filter. But Abdalla et. al [ABC+08] noted that the BDOP filter could lack *consistency*, meaning turn up false positives. Their solution was to modify the construction. What we observed instead was that consistency would in fact be provided by the *original* construct if the IBE scheme was robust. PEKS consistency turns out to correspond exactly to communication correctness of the anonymous IBE scheme in the sense discussed above. (Because the PEKS messages in the BDOP scheme are the recipients identities from the IBE perspective.) Besides resurrecting the BDOP construct, the robustness approach allows us to obtain the first consistent IND-CCA secure PEKS without random oracles.

Sako's auction protocol [Sak00] is important because it was the first truly practical one to hide the bids of losers. It makes clever use of anonymous encryption for privacy. But we present an attack on fairness whose cause is ultimately a lack of robustness in the anonymous encryption scheme (cf. Appendix G.8).

All this underscores a number of the claims we are making about robustness: that it is of conceptual value; that it makes encryption more resistant to mis-use; that it has been implicitly (and incorrectly) assumed; and that there is value to making it explicit, formally defining and provably achieving it.

WEAK VERSUS STRONG. The above-mentioned auction protocol fails because an adversary can create a ciphertext that decrypts correctly under any decryption key. Strong robustness is needed to prevent this. Weak robustness (of the underlying IBE) will yield PEKS consistency for honestly-encrypted messages but may allow spammers to bypass all filters with a single ciphertext, something prevented by strong robustness. Strong robustness trumps weak for applications and goes farther towards making encryption mis-use resistant. We have defined and

considered the weaker version because it can be more efficiently achieved, because some existing schemes achieve it and because attaining it is a crucial first step in our method for attaining strong robustness.

ACHIEVING ROBUSTNESS. As the reader has surely already noted, robustness (even strong) is trivially achieved by appending the encryption key to the ciphertext and checking for it upon decryption. The problem is that the resulting scheme is not anonymous and, as we have seen above, it is exactly for anonymous schemes that robustness is important. Of course, data privacy is important too. Letting AI-ATK = ANO-ATK + IND-ATK for ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$, our goal is to achieve AI-ATK + XROB-ATK, ideally for both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$ and X $\in \{\mathrm{W}, \mathrm{S}\}$. This is harder.

TRANSFORMS. It is natural to begin by seeking a general transform that takes an arbitrary AI-ATK scheme and returns a AI-ATK + XROB-ATK one. This allows us to exploit known constructions of AI-ATK schemes, supports modular protocol design and also helps understand robustness divorced from the algebra of specific schemes. Furthermore, there is a natural and promising transform to consider. Namely, before encrypting, append to the message some redundancy, such as the recipient encryption key, a constant, or even a hash of the message, and check for its presence upon decryption. (Adding the redundancy before encrypting rather than after preserves AI-ATK.) Intuitively this should provide robustness because decryption with the "wrong" key will result, if not in rejection, then in recovery of a garbled plaintext, unlikely to possess the correct redundancy.

The truth is more complex. We consider two versions of the paradigm and summarize our findings in Figure G.1. In encryption with *unkeyed redundancy*, the redundancy is a function RC of the message and encryption key alone. In this case we show that the method fails spectacularly, not providing even *weak* robustness *regardless of the choice of the function* RC. In encryption with *keyed redundancy*, we allow RC to depend on a key K that is placed in the public parameters of the transformed scheme, out of direct reach of the algorithms of the original scheme. In this form, the method can easily provide weak robustness, and that too with a very simple redundancy function, namely the one that simply returns K.

But we show that even encryption with keyed redundancy fails to provide *strong* robustness. To achieve the latter we have to step outside the encryption with redundancy paradigm. We present a strong robustness conferring transform that uses a (non-interactive) commitment scheme. For subtle reasons, for this transform to work the starting scheme needs to already be weakly robust. If it isn't already, we can make it so via our weak robustness transform.

In summary, on the positive side we provide a transform conferring weak robustness and another conferring strong robustness. Given any AI-ATK scheme the first transform returns a WROB-ATK + AI-ATK one. Given any AI-ATK + WROB-ATK scheme the second transform returns a SROB-ATK + AI-ATK one. In both cases it is for both ATK = CPA and ATK = CCA and in both cases the transform applies to what we call general encryption schemes, of which both PKE and IBE are special cases, so both are covered.

ROBUSTNESS OF SPECIFIC SCHEMES. The robustness of existing schemes is important because they might be in use. We ask which specific existing schemes are robust, and, for those that are not, whether they can be made so at a cost lower than that of applying one of our general transforms. There is no reason to expect schemes that are only AI-CPA to be robust since the decryption algorithm may never reject, so we focus on schemes that are known to be AI-CCA. This narrows the field quite a bit. Our findings and results are summarized in Figure G.1.

Canonical AI-CCA schemes in the PKE setting are Cramer-Shoup ($\mathcal{CS}$) in the standard model [CS03, BBDP01] and $\mathcal{DHIES}$ in the random oracle (RO) model [ABR01, BBDP01]. We show that both are WROB-CCA but neither is SROB-CCA, the latter because encryption with

Transform	WROB-ATK	SROB-ATK
Encryption with unkeyed redundancy (EuR)	No	No
Encryption with keyed redundancy (EkR)	Yes	No

Scheme	**setting**	AI-CCA	WROB-CCA	SROB-CCA	**RO model**
$\mathcal{CS}$	PKE	Yes [CS03, BBDP01]	Yes	No	No
$\mathcal{CS}^*$	PKE	Yes	Yes	Yes	No
$\mathcal{DHIES}$	PKE	Yes [ABR01]	Yes	No	Yes
$\mathcal{DHIES}^*$	PKE	Yes	Yes	Yes	Yes
$\mathcal{BF}$	IBE	Yes [BF01, ABC$^+$08]	Yes	Yes	Yes
$\mathcal{BW}$	IBE	Yes [BW06]	No	No	No

Figure G.1: **Achieving Robustness.** The first table summarizes our findings on the encryption with redundancy transform. "No" means the method fails to achieve the indicated robustness for *all* redundancy functions, while "yes" means there exists a redundancy function for which it works. The second table summarizes robustness results about some specific AI-CCA schemes.

0 randomness yields a ciphertext valid under any encryption key. We present modified versions $\mathcal{CS}^*, \mathcal{DHIES}^*$ of the schemes that we show are SROB-CCA. Our proof that $\mathcal{CS}^*$ is SROB-CCA builds on the information-theoretic part of the proof of [CS03]. The result does not need to assume hardness of DDH. It relies instead on pre-image security of the underlying hash function for random range points, something not implied by collision-resistance but seemingly possessed by candidate functions.

In the IBE setting, the CCA version $\mathcal{BF}$ of the RO model Boneh-Franklin scheme is AI-CCA [BF01, ABC$^+$08], and we show it is SROB-CCA. The standard model Boyen-Waters scheme $\mathcal{BW}$ is AI-CCA [BW06], and we show it is neither WROB-CCA nor SROB-CCA. It can be made either via our transforms but we don't know of any more direct way to do this.

$\mathcal{BF}$ is obtained via the Fujisaki-Okamoto (FO) transform [FO99] and $\mathcal{BW}$ via the Canetti-Halevi-Katz (CHK) transform [CHK04, BCHK07]. We can show that neither transform *generically* provides strong robustness. This doesn't say whether they do or not when applied to specific schemes, and indeed the first does for $\mathcal{BF}$ and the second does not for $\mathcal{BW}$.

Summary. Protocol design suggests that designers have the intuition that robustness is naturally present. This seems to be more often right than wrong when considering *weak* robustness of *specific* AI-CCA schemes. Prevailing intuition about *generic* ways to add even weak robustness is wrong, yet we show it can be done by an appropriate tweak of these ideas. Strong robustness is more likely to be absent than present in specific schemes, but important schemes can be patched. Strong robustness can also be added generically, but with more work.

Related work. There is growing recognition that robustness is important in applications and worth defining explicitly, supporting our own claims to this end. In particular the correctness requirement for predicate encryption [KSW08] includes a form of weak robustness and, in recent work concurrent to, and independent of, ours, Hofheinz and Weinreb [HW08] introduced a notion of *well-addressedness* of IBE schemes that is just like weak robustness except that the adversary gets the IBE master secret key. Neither work considers or achieves strong robustness, and neither treats PKE.

proc Initialize
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$; $b \xleftarrow{\$} \{0,1\}$
$S, T, U, V \leftarrow \emptyset$
Return $pars$

proc GetEK(id)
$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

proc GetDK(id)
If $id \notin U$ then return $\bot$
If $id \in S$ then return $\bot$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

proc Dec(C, id)
If $id \notin U$ then return $\bot$
If $(id, C) \in T$ then return $\bot$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
Return M

proc LR$(id_0^*, id_1^*, M_0^*, M_1^*)$
If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\bot$
If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\bot$
If $|M_0^*| \neq |M_1^*|$ then return $\bot$
$C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b], M_b^*)$
$S \leftarrow S \cup \{id_0^*, id_1^*\}$
$T \leftarrow T \cup \{(id_0^*, C^*), (id_1^*, C^*)\}$
Return C^*

proc Finalize(b')
Return $(b' = b)$

Figure G.2: Game $\mathrm{AI}_{\mathcal{GE}}$ defining AI-ATK security of general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$.

G.2 Definitions

Notation and conventions. If x is a string then $|x|$ denotes its length, and if S is a set then $|S|$ denotes its size. The empty string is denoted ε. By $a_1 \| \ldots \| a_n$, we denote a string encoding of $a_1, \ldots, a_n$ from which $a_1, \ldots, a_n$ are uniquely recoverable. (Usually, concatenation suffices.) By $a_1 \| \ldots \| a_n \leftarrow a$, we mean that a is parsed into its constituents $a_1, \ldots, a_n$. Similarly, if $a = (a_1, \ldots, a_n)$ then $(a_1, \ldots, a_n) \leftarrow a$ means we parse a as shown. Unless otherwise indicated, an algorithm may be randomized. By $y \xleftarrow{\$} A(x_1, x_2, \ldots)$ we denote the operation of running A on inputs $x_1, x_2, \ldots$ and fresh coins and letting y denote the output. We denote by $[A(x_1, x_2, \ldots)]$ the set of all possible outputs of A on inputs $x_1, x_2, \ldots$. We assume that an algorithm returns $\bot$ if any of its inputs is $\bot$.

Games. Our definitions and proofs use code-based game-playing [BR06]. Recall that a game —look at Figure G.2 for an example— has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game G is executed with an adversary A as follows. First, **Initialize** executes and its outputs are the inputs to A. Then A executes, its oracle queries being answered by the corresponding procedures of G. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter, denoted G^A, is called the output of the game, and we let "G^A" denote the event that this game output takes value true. Boolean flags are assumed initialized to false. Games $\mathrm{G}_i, \mathrm{G}_j$ are *identical until* bad if their code differs only in statements that follow the setting of bad to true. Our proofs will use the following.

Lemma G.2.1 [BR06] Let $\mathrm{G}_i, \mathrm{G}_j$ be identical until bad games, and A an adversary. Then

$$\left|\Pr\left[\mathrm{G}_i^A\right] - \Pr\left[\mathrm{G}_j^A\right]\right| \leq \Pr\left[\mathrm{G}_j^A \text{ sets } \mathsf{bad}\right] . \blacksquare$$

The running time of an adversary is the worst case time of the execution of the adversary with the game defining its security, so that the execution time of the called game procedures is included.

General encryption. We introduce and use general encryption schemes, of which both PKE and IBE are special cases. This allows us to avoid repeating similar definitions and proofs. A *general encryption* (GE) scheme is a tuple $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of algorithms. The

proc Initialize
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$; $U, V \leftarrow \emptyset$
Return $pars$

proc GetEK(id)
$U \leftarrow U \cup \{id\}$
$(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return $\mathsf{EK}[id]$

proc GetDK(id)
If $id \notin U$ then return $\bot$
$V \leftarrow V \cup \{id\}$
Return $\mathsf{DK}[id]$

proc Dec(C, id)
If $id \notin U$ then return $\bot$
$M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
Return M

proc Finalize(M, id_0, id_1) // $\mathrm{WROB}_{\mathcal{GE}}$
If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$M_0 \leftarrow M$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], M_0)$
$M_1 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
Return $(M_0 \neq \bot) \wedge (M_1 \neq \bot)$

proc Finalize(C, id_0, id_1) // $\mathrm{SROB}_{\mathcal{GE}}$
If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
If $(id_0 \in V) \vee (id_1 \in V)$ then return false
If $(id_0 = id_1)$ then return false
$M_0 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_0], \mathsf{DK}[id_0], C)$
$M_1 \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
Return $(M_0 \neq \bot) \wedge (M_1 \neq \bot)$

Figure G.3: Games $\mathrm{WROB}_{\mathcal{GE}}$ and $\mathrm{SROB}_{\mathcal{GE}}$ defining WROB-ATK and SROB-ATK security (respectively) of general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. The procedures on the left are common to both games, which differ only in their **Finalize** procedures.

parameter generation algorithm PG takes no input and returns common parameter $pars$ and a master secret key msk. On input $pars, msk, id$, the key generation algorithm KG produces an encryption key ek and decryption key dk. On inputs $pars, ek, M$, the encryption algorithm Enc produces a ciphertext C encrypting plaintext M. On input $pars, ek, dk, C$, the deterministic decryption algorithm Dec returns either a plaintext message M or $\bot$ to indicate that it rejects. We say that $\mathcal{GE}$ is a public-key encryption (PKE) scheme if $msk = \varepsilon$ and KG ignores its id input. To recover the usual syntax we may in this case write the output of PG as $pars$ rather than $(pars, msk)$ and omit msk, id as inputs to KG. We say that $\mathcal{GE}$ is an identity-based encryption (IBE) scheme if $ek = id$, meaning the encryption key created by KG on inputs $pars, msk, id$ always equals id. To recover the usual syntax we may in this case write the output of KG as dk rather than (ek, dk). It is easy to see that in this way we have recovered the usual primitives. But there are general encryption schemes that are neither PKE nor IBE schemes, meaning the primitive is indeed more general.

Correctness. Correctness of a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ requires that, for all $(pars, msk) \in [\mathsf{PG}]$, all plaintexts M in the underlying message space associated to $pars$, all identities id, and all $(ek, dk) \in [\mathsf{KG}(pars, msk, id)]$, we have $\mathsf{Dec}(pars, ek, dk, \mathsf{Enc}(pars, ek, M)) = M$ with probability one, where the probability is taken over the coins of Enc.

AI-ATK security. Historically, definitions of data privacy (IND) [GM84, RS92, DDN00, BDPR98, BF03] and anonymity (ANON) [BBDP01, ABC+08] have been separate. We are interested in schemes that achieve both, so rather than use separate definitions we follow [BGH07] and capture both simultaneously via game $\mathrm{AI}_{\mathcal{GE}}$ of Figure G.2. A cpa adversary is one that makes no **Dec** queries, and a cca adversary is one that might make such queries. The ai-advantage of such an adversary, in either case, is

$$\mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(A) = 2 \cdot \Pr\left[\, \mathrm{AI}^{A}_{\mathcal{GE}} \,\right] - 1.$$

We will assume an ai-adversary makes only one **LR** query, since a hybrid argument shows that making q of them can increase its ai-advantage by a factor of at most q.

Oracle **GetDK** represents the IBE key-extraction oracle [BF03]. In the PKE case it is

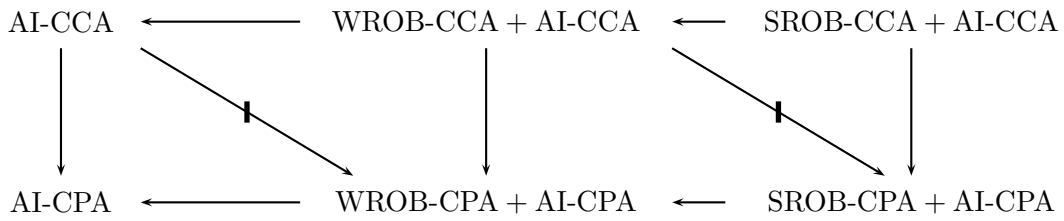


Figure G.4: **Relations between notions.** An arrow A $\rightarrow$ B is an implication, meaning every scheme that is A-secure is also B-secure, while a barred arrow A $\not\rightarrow$ B is a separation, meaning that there is a A-secure scheme that is not B-secure. (Assuming of course that there exists a A-secure scheme in the first place.)

superfluous in the sense that removing it results in a definition that is equivalent up to a factor depending on the number of **GetDK** queries. That's probably why the usual definition has no such oracle. But conceptually, if it is there for IBE, it ought to be there for PKE, and it does impact concrete security.

Robustness. Associated to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are games WROB, SROB of Figure G.3. As before, a cpa adversary is one that makes no **Dec** queries, and a cca adversary is one that might make such queries. The wrob and srob advantages of an adversary, in either case, are

$$\mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(A) = \Pr\left[\,\mathrm{WROB}^{A}_{\mathcal{GE}}\,\right] \qquad \text{and} \qquad \mathbf{Adv}^{\mathrm{srob}}_{\mathcal{GE}}(A) = \Pr\left[\,\mathrm{SROB}^{A}_{\mathcal{GE}}\,\right].$$

The difference between WROB and SROB is that in the former the adversary produces a message M, and C is its encryption under the encryption key of one of the given identities, while in the latter it produces C directly, and may not obtain it as an honest encryption. It is worth clarifying that in the PKE case the adversary does *not* get to choose the encryption (public) keys of the identities it is targeting. These are honestly and independently chosen, in real life by the identities themselves and in our formalization by the games.

Relations between notions. Figure G.4 shows implications and separations in the style of [BDPR98]. We consider each robustness notion in conjunction with the corresponding AI one since robustness is interesting only in this case. The implications are all trivial. The first separation shows that the strongest notion of privacy fails to imply even the weakest type of robustness. The second separation shows that weak robustness, even under CCA, doesn't imply strong robustness. We stress that here an implication A $\rightarrow$ B means that any A-secure, *unaltered*, is B-secure. Correspondingly, a non-implication A $\not\rightarrow$ B means that there is an A-secure that, unaltered, is not B-secure. (It doesn't mean that an A-secure scheme can't be transformed into a B-secure one.) Only a minimal set of arrows and barred arrows is shown; others can be inferred. The picture is complete in the sense that it implies either an implication or a separation between any pair of notions.

G.3 Robustness failures of encryption with redundancy

A natural privacy-and-anonymity-preserving approach to add robustness to an encryption scheme is to add redundancy before encrypting, and upon decryption reject if the redundancy is absent. Here we investigate the effectiveness of this encryption with redundancy approach, justifying the negative results discussed in Section G.1 and summarized in the first table of Figure G.1.

RKG	$\mathsf{RC}(K, ek\|M)$	$\mathsf{RV}(K, ek\|M, r)$
Return $K \leftarrow \varepsilon$	Return ε	Return 1
Return $K \leftarrow \varepsilon$	Return 0^k	Return $(r = 0^k)$
Return $K \leftarrow \varepsilon$	Return ek	Return $(r = ek)$
Return $K \leftarrow \varepsilon$	$L \xleftarrow{\$} \{0,1\}^k$; Return $L\|H(L, ek\|M)$	$L\|h \leftarrow r$; Return $(h = H(L, ek\|M))$
Return $K \xleftarrow{\$} \{0,1\}^k$	Return K	Return $(r = K)$
Return $K \xleftarrow{\$} \{0,1\}^k$	Return $H(K, ek\|M)$	Return $(r = H(K, ek\|M))$

Figure G.5: Examples of redundancy codes, where the data x is of the form $ek\|M$. The first four are unkeyed and the last two are keyed.

Redundancy codes and the transform. A redundancy code $\mathcal{RED} = (\mathsf{RKG}, \mathsf{RC}, \mathsf{RV})$ is a triple of algorithms. The redundancy key generation algorithm RKG generates a key K. On input K and data x the redundancy computation algorithm RC returns redundancy r. Given K, x, and claimed redundancy r, the deterministic redundancy verification algorithm RV returns 0 or 1. We say that $\mathcal{RED}$ is unkeyed if the key K output by RKG is always equal to ε, and keyed otherwise. The correctness condition is that for all x we have $\mathsf{RV}(K, x, \mathsf{RC}(K, x)) = 1$ with probability one, where the probability is taken over the coins of RKG and RC. (We stress that the latter is allowed to be randomized.)

Given a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a redundancy code $\mathcal{RED} = (\mathsf{RKG}, \mathsf{RC}, \mathsf{RV})$, the *encryption with redundancy transform* associates to them the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are shown on the left side of Figure G.6. Note that the transform has the first of our desired properties, namely that it preserves AI-ATK. Also if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$, and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, which means the results we obtain here apply to both settings.

Figure G.5 shows example redundancy codes for the transform. With the first, $\overline{\mathcal{GE}}$ is identical to $\mathcal{GE}$, so that the counterexample below shows that AI-CCA does not imply WROB-CPA , justifying the first separation of Figure G.4.The second and third rows show redundancy equal to a constant or the encryption key as examples of (unkeyed) redundancy codes. The fourth row shows a code that is randomized but still unkeyed. The hash function H could be a MAC or a collision resistant function. The last two are keyed redundancy codes, the first the simple one that just always returns the key, and the second using a hash function. Obviously, there are many other examples.

SROB failure. We show that encryption with redundancy fails to provide strong robustness for *all* redundancy codes, whether keyed or not. More precisely, we show that for any redundancy code $\mathcal{RED}$ and both ATK $\in \{\text{CPA}, \text{CCA}\}$, there is an AI-ATK encryption scheme $\mathcal{GE}$ such that the scheme $\overline{\mathcal{GE}}$ resulting from the encryption-with-redundancy transform applied to $\mathcal{GE}, \mathcal{RED}$ is not SROB-CPA. We build $\mathcal{GE}$ by modifying a given AI-ATK encryption scheme $\mathcal{GE}^* = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}^*, \mathsf{Dec}^*)$. Let l be the number of coins used by RC, and let $\mathsf{RC}(x; \omega)$ denote the result of executing RC on input x with coins $\omega \in \{0,1\}^l$. Let M^* be a function that given *pars* returns a point in the message space associated to *pars* in $\mathcal{GE}^*$. Then $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ where the new algorithms are shown on the bottom right side of Figure G.6. The reason we used 0^l as coins for RC here is that Dec is required to be deterministic.

Our first claim is that the assumption that $\mathcal{GE}^*$ is AI-ATK implies that $\mathcal{GE}$ is too. Our second claim, that $\overline{\mathcal{GE}}$ is not SROB-CPA, is demonstrated by the following attack. For a pair id_0, id_1 of distinct identities of its choice, the adversary A, on input $(pars, K)$, begins with queries $ek_0 \xleftarrow{\$} \mathbf{GetEK}(id_0)$ and $ek_1 \xleftarrow{\$} \mathbf{GetEK}(id_1)$. It then creates ciphertext $C \leftarrow 0 \,\|\, K$ and returns (id_0, id_1, C). We claim that $\mathbf{Adv}^{\text{srob}}_{\overline{\mathcal{GE}}}(A) = 1$. Letting dk_0, dk_1 denote the decryption keys

Algorithm $\overline{\mathsf{PG}}$
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$; $K \xleftarrow{\$} \mathsf{RKG}$
Return $((pars, K), msk)$

Algorithm $\overline{\mathsf{KG}}((pars, K), msk, id)$
$(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return ek

Algorithm $\overline{\mathsf{Enc}}((pars, K), ek, M)$
$r \xleftarrow{\$} \mathsf{RC}(K, ek \| M)$
$C \xleftarrow{\$} \mathsf{Enc}(pars, ek, M \| r)$
Return C

Algorithm $\overline{\mathsf{Dec}}((pars, K), ek, dk, C)$
$M \| r \leftarrow \mathsf{Dec}(pars, ek, dk, C)$
If $\mathsf{RV}(K, ek \| M, r) = 1$ then return M
Else return $\perp$

Algorithm $\mathsf{Enc}(pars, ek, M)$
$C \xleftarrow{\$} \mathsf{Enc}^*(pars, ek, M)$
Return C

Algorithm $\mathsf{Dec}(pars, ek, dk, C)$
$M \leftarrow \mathsf{Dec}^*(pars, ek, dk, C)$
If $M = \perp$ then
  $M \leftarrow M^*(pars) \| \mathsf{RC}(\varepsilon, ek \| M^*(pars); 0^l)$
Return M

Algorithm $\mathsf{Enc}(pars, ek, M)$
$C^* \xleftarrow{\$} \mathsf{Enc}^*(pars, ek, M)$
Return $1 \| C^*$

Algorithm $\mathsf{Dec}(pars, ek, dk, C)$
$b \| C^* \leftarrow C$
If $b = 1$ then return $\mathsf{Dec}^*(pars, ek, dk, C^*)$
Else return $M^*(pars) \| \mathsf{RC}(C^*, ek \| M^*(pars); 0^l)$

Figure G.6: **Left**: Transformed scheme for the encryption with redundancy paradigm. **Top Right**: Counterexample for WROB. **Bottom Right**: Counterexample for SROB.

corresponding to ek_0, ek_1 respectively, the reason is the following. For both $b \in \{0, 1\}$, the output of $\mathsf{Dec}(pars, ek_b, dk_b, C)$ is $M^*(pars) \| r_b(pars)$ where $r_b(pars) = \mathsf{RC}(K, ek_b \| M^*(pars); 0^l)$. But the correctness of $\mathcal{RED}$ implies that $\mathsf{RV}(K, ek_b \| M^*(pars), r_b(pars)) = 1$ and hence $\overline{\mathsf{Dec}}((pars, K), ek_b, dk_b, C)$ returns $M^*(pars)$ rather than $\perp$.

WROB FAILURE. We show that encryption with redundancy fails to provide even *weak* robustness for all *unkeyed* redundancy codes. This is still a powerful negative result because many forms of redundancy that might intuitively work, such the first four of Figure G.5, are included. More precisely, we claim that for any unkeyed redundancy code $\mathcal{RED}$ and both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$, there is an AI-ATK encryption scheme $\mathcal{GE}$ such that the scheme $\overline{\mathcal{GE}}$ resulting from the encryption-with-redundancy transform applied to $\mathcal{GE}, \mathcal{RED}$ is not WROB-CPA. We build $\mathcal{GE}$ by modifying a given AI-ATK + WROB-CPA encryption scheme $\mathcal{GE}^* = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}^*, \mathsf{Dec}^*)$. With notation as above, the new algorithms for the scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ are shown on the top right side of Figure G.6.

Our first claim is that the assumption that $\mathcal{GE}^*$ is AI-ATK implies that $\mathcal{GE}$ is too. Our second claim, that $\overline{\mathcal{GE}}$ is not WROB-CPA, is demonstrated by the following attack. For a pair id_0, id_1 of distinct identities of its choice, the adversary A, on input $(pars, \varepsilon)$, makes queries $ek_0 \xleftarrow{\$} \mathbf{GetEK}(id_0)$ and $ek_1 \xleftarrow{\$} \mathbf{GetEK}(id_1)$ and returns $(id_0, id_1, M^*(pars))$. We claim that $\mathbf{Adv}^{\mathrm{wrob}}_{\overline{\mathcal{GE}}}(A)$ is high. Letting dk_1 denote the decryption key corresponding to ek_1, the reason is the following. Let $r_0 \xleftarrow{\$} \mathsf{RC}(\varepsilon, ek_0 \| M^*(pars))$ and $C \xleftarrow{\$} \mathsf{Enc}(pars, ek_0, M^*(pars) \| r_0)$. The assumed WROB-CPA security of $\mathcal{GE}^*$ implies that $\mathsf{Dec}(pars, ek_1, dk_1, C)$ is most probably $M^*(pars) \| r_1(pars)$ where $r_1(pars) = \mathsf{RC}(\varepsilon, ek_1 \| M^*(pars); 0^l)$. But the correctness of $\mathcal{RED}$ implies that $\mathsf{RV}(\varepsilon, ek_1 \| M^*(pars), r_1(pars)) = 1$ and hence $\overline{\mathsf{Dec}}((pars, \varepsilon), ek_1, dk_1, C)$ returns $M^*(pars)$ rather than $\perp$.

G.4 Transforms that work

We present a transform that confers weak robustness and another that confers strong robustness. They preserve privacy and anonymity, work for PKE as well as IBE, and for CPA as well as

CCA. In both cases the security proofs surface some delicate issues. Besides being useful in its own right, the weak robustness transform is a crucial step in obtaining strong robustness, so we begin there.

WEAK ROBUSTNESS TRANSFORM. We saw that encryption-with-redundancy fails to provide even weak robustness if the redundancy code is unkeyed. Here we show that if the redundancy code is keyed, even in the simplest possible way where the redundancy is just the key itself, the transform does provide weak robustness, turning any AI-ATK secure general encryption scheme into an AI-ATK + WROB-ATK one, for both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$.

The transformed scheme encrypts with the message a key K placed in the public parameters. In more detail, the *weak robustness transform* associates to a given general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and integer parameter k, representing the length of K, the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are depicted in Figure G.7. Note that if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$ and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, so that our results, captured by Theorem G.4.1 below, cover both settings.

The intuition for the weak robustness of $\overline{\mathcal{GE}}$ is that the $\mathcal{GE}$ decryption under one key, of an encryption of $\overline{M}\|K$ created under another key, cannot, by the assumed AI-ATK security of $\mathcal{GE}$, reveal K, and hence the check will fail. This is pretty much right for PKE, but the delicate issue is that for IBE, information about K can enter via the identities, which in this case are the encryption keys and are chosen by the adversary as a function of K. The AI-ATK security of $\mathcal{GE}$ is no protection against this. We show however that this can be dealt with by making K sufficiently longer than the identities.

Theorem G.4.1 Let $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme with identity space $\{0,1\}^n$, and let $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be the general encryption scheme resulting from applying the weak robustness transform to $\mathcal{GE}$ and integer parameter k. Then

1. AI-ATK: Let A be an ai-adversary against $\overline{\mathcal{GE}}$. Then there is an ai-adversary B against $\mathcal{GE}$ such that

$$\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) \;=\; \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B)\ .$$

Adversary B inherits the query profile of A and has the same running time as A. If A is a cpa adversary then so is B.

2. WROB-ATK: Let A be a wrob adversary against $\overline{\mathcal{GE}}$ with running time t, and let $\ell = 2n + \lceil \log_2(t) \rceil$. Then there is an ai-adversary B against $\mathcal{GE}$ such that

$$\mathbf{Adv}^{\mathrm{wrob}}_{\overline{\mathcal{GE}}}(A) \;\le\; \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) + 2^{\ell-k}\ .$$

Adversary B inherits the query profile of A and has the same running time as A. If A is a cpa adversary then so is B. ▮

The first part of the theorem implies that if $\mathcal{GE}$ is AI-ATK then $\overline{\mathcal{GE}}$ is AI-ATK as well. The second part of the theorem implies that if $\mathcal{GE}$ is AI-ATK and k is chosen sufficiently larger than $2n + \lceil \log_2(t) \rceil$ then $\overline{\mathcal{GE}}$ is WROB-ATK. In both cases this is for both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$. The theorem says it directly for CCA, and for CPA by the fact that if A is a cpa adversary then so is B. When we say that B inherits the query profile of A we mean that for every oracle that B has, if A has an oracle of the same name and makes q queries to it, then this is also the number B makes. The proof of the first part of the theorem is straightforward and is omitted. The proof of the second part is given in Appendix G.9. It is well known that collision-resistant hashing of identities preserves AI-ATK and serves to make them of fixed length [BB04a] so the assumption that the identity space is $\{0,1\}^n$ rather than $\{0,1\}^*$ is not really a restriction. In practice we might hash with SHA256 so that $n = 256$, and, assuming $t \le 2^{128}$, setting $k = 768$ would make $2^{\ell-k} = 2^{-128}$.

Algorithm $\overline{\mathsf{PG}}$
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$
$K \xleftarrow{\$} \{0,1\}^k$
Return $((pars, K), msk)$

Algorithm $\overline{\mathsf{Enc}}((pars, K), ek, \overline{M})$
$C \xleftarrow{\$} \mathsf{Enc}(pars, ek, \overline{M} \| K))$
Return C

Algorithm $\overline{\mathsf{KG}}((pars, K), msk, id)$
$(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return (ek, dk)

Algorithm $\overline{\mathsf{Dec}}((pars, K), ek, dk, C)$
$M \leftarrow \mathsf{Dec}(pars, ek, dk, C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| K^* \leftarrow M$
If $(K = K^*)$ then return $\overline{M}$
Else Return $\perp$

Figure G.7: General encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ resulting from applying our weak-robustness transform to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and integer parameter k.

COMMITMENT SCHEMES. Our strong robustness transform will use commitments. A commitment scheme is a 3-tuple $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$. The parameter generation algorithm CPG returns public parameters *cpars*. The committal algorithm Com takes *cpars* and data x as input and returns a commitment *com* to x along with a decommittal key *dec*. The deterministic verification algorithm Ver takes *cpars*, x, *com*, *dec* as input and returns 1 to indicate that accepts or 0 to indicate that it rejects. Correctness requires that, for any $x \in \{0,1\}^*$, any $cpars \in [\mathsf{CPG}]$, and any $(com, dec) \in [\mathsf{Com}(cpars, x)]$, we have that $\mathsf{Ver}(cpars, x, com, dec) = 1$ with probability one, where the probability is taken over the coins of Com. We require the scheme to have the *uniqueness* property, which means that for any $x \in \{0,1\}^*$, any $cpars \in [\mathsf{CPG}]$, and any $(com, dec) \in [\mathsf{Com}(cpars, x)]$ it is the case that $\mathsf{Ver}(cpars, x, com^*, dec) = 0$ for all $com^* \neq com$. In most schemes the decommittal key is the randomness used by the committal algorithm and verification is by re-applying the committal function, which ensures uniqueness. The advantage measures $\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A)$ and $\mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(A)$, referring to the standard hiding and binding properties, are recalled in Appendix G.6. We refer to the corresponding notions as HIDE and BIND.

THE STRONG ROBUSTNESS TRANSFORM. The idea is for the ciphertext to include a commitment to the encryption key. The commitment is *not* encrypted, but the decommittal key is. In detail, given a general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and a commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$ the *strong robustness transform* associates to them the general encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ whose algorithms are depicted in Figure G.8. Note that if $\mathcal{GE}$ is a PKE scheme then so is $\overline{\mathcal{GE}}$ and if $\mathcal{GE}$ is an IBE scheme then so is $\overline{\mathcal{GE}}$, so that our results, captured by the Theorem G.4.2, cover both settings.

In this case the delicate issue is not the robustness but the AI-ATK security of $\overline{\mathcal{GE}}$ in the CCA case. Intuitively, the hiding security of the commitment scheme means that a $\overline{\mathcal{GE}}$ ciphertext does not reveal the encryption key. As a result, we would expect AI-ATK security of $\overline{\mathcal{GE}}$ to follow from the commitment hiding security and the assumed AI-ATK security of $\mathcal{GE}$. This turns out not to be true, and demonstrably so, meaning there is a counterexample to this claim. (See below.) What we show is that the claim is true if $\mathcal{GE}$ is additionally WROB-ATK. This property, if not already present, can be conferred by first applying our weak robustness transform.

Theorem G.4.2 Let $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be a general encryption scheme, and let $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be the general encryption scheme resulting from applying the strong robustness transform to $\mathcal{GE}$ and commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$. Then

1. AI-ATK: Let A be an ai-adversary against $\overline{\mathcal{GE}}$. Then there is a wrob adversary W against

Algorithm $\overline{\mathsf{PG}}$
$(pars, msk) \xleftarrow{\$} \mathsf{PG}$
$cpars \xleftarrow{\$} \mathsf{CPG}$
Return $((pars, cpars), msk)$

Algorithm $\overline{\mathsf{Enc}}((pars, cpars), ek, \overline{M})$
$(com, dec) \xleftarrow{\$} \mathsf{Com}(cpars, ek)$
$C \xleftarrow{\$} \mathsf{Enc}(pars, ek, \overline{M} \| dec))$
Return (C, com)

Algorithm $\overline{\mathsf{KG}}((pars, cpars), msk, id)$
$(ek, dk) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
Return (ek, dk)

Algorithm $\overline{\mathsf{Dec}}((pars, cpars), ek, dk, (C, com))$
$M \leftarrow \mathsf{Dec}(pars, ek, dk, C)$
If $M = \perp$ then return $\perp$
$\overline{M} \| dec \leftarrow M$
If $(\mathsf{Ver}(cpars, ek, com, dec) = 1)$ then return $\overline{M}$
Else Return $\perp$

Figure G.8: General encryption scheme $\overline{\mathcal{GE}} = (\overline{\mathsf{PG}}, \overline{\mathsf{KG}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ resulting from applying our strong robustness transform to general encryption scheme $\mathcal{GE} = (\mathsf{PG}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and commitment scheme $\mathcal{CMT} = (\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$.

$\mathcal{GE}$, a hiding adversary H against $\mathcal{CMT}$ and an ai-adversary B against $\mathcal{GE}$ such that

$$\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) \;\le\; 2 \cdot \mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(W) + 2 \cdot \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(H) + 3 \cdot \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) \;.$$

Adversaries W, B inherit the query profile of A, and adversaries W, H, B have the same running time as A. If A is a cpa adversary then so are W, B.

2. SROB-ATK: Let A be a srob adversary against $\overline{\mathcal{GE}}$ making q **GetEK** queries. Then there is a binding adversary B against $\mathcal{CMT}$ such that

$$\mathbf{Adv}^{\mathrm{srob}}_{\overline{\mathcal{GE}}}(A) \;\le\; \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(B) + \binom{q}{2} \cdot \mathbf{Coll}_{\mathcal{GE}} \;.$$

Adversary B has the same running time as A. ▮

The first part of the theorem implies that if $\mathcal{GE}$ is AI-ATK and WROB-ATK and $\mathcal{CMT}$ is HIDE then $\overline{\mathcal{GE}}$ is AI-ATK, and the second part of the theorem implies that if $\mathcal{CMT}$ is BIND secure and $\mathcal{GE}$ has low encryption key collision probability then $\overline{\mathcal{GE}}$ is SROB-ATK. In both cases this is for both ATK $\in \{\mathrm{CPA}, \mathrm{CCA}\}$. We remark that the proof shows that in the CPA case the WROB-ATK assumption on $\mathcal{GE}$ in the first part is actually not needed. The encryption key collision probability $\mathbf{Coll}_{\mathcal{GE}}$ of $\mathcal{GE}$ is defined as the maximum probability that $ek_0 = ek_1$ in the experiment

$$(pars, msk) \xleftarrow{\$} \mathsf{PG} \,;\; (ek_0, dk_0) \xleftarrow{\$} \mathsf{KG}(pars, msk, id_0) \,;\; (ek_1, dk_1) \xleftarrow{\$} \mathsf{KG}(pars, msk, id_1) \;,$$

where the maximum is over all distinct identities id_0, id_1. The collision probability is zero in the IBE case since $ek_0 = id_0 \neq id_1 = ek_1$. It is easy to see that $\mathcal{GE}$ being AI implies $\mathbf{Coll}_{\mathcal{GE}}$ is negligible, so asking for low encryption key collision probability is in fact not an extra assumption. (For a general encryption scheme the adversary needs to have hardwired the identities that achieve the maximum, but this is not necessary for PKE because here the probability being maximized is the same for all pairs of distinct identities.) The reason we made the encryption key collision probability explicit is that for most schemes it is unconditionally low. For example, when $\mathcal{GE}$ is the ElGamal PKE scheme, it is $1/|\mathbb{G}|$ where $\mathbb{G}$ is the group being used. Proofs of both parts of the theorem are in Appendix G.9.

The need for weak-robustness. As we said above, the AI-ATK security of $\overline{\mathcal{GE}}$ won't be implied merely by that of $\mathcal{GE}$. (We had to additionally assume that $\mathcal{GE}$ is WROB-ATK.) Here we justify this somewhat counter-intuitive claim. This discussion is informal but can be turned into a formal counterexample. Imagine that the decryption algorithm of $\mathcal{GE}$ returns a fixed string of the form $(\hat{M}, \hat{dec})$ whenever the wrong key is used to decrypt. Moreover, imagine $\mathcal{CMT}$ is such

Algorithm PG
$K \xleftarrow{\$} \mathsf{Keys}(H)$; $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$
$g_2 \leftarrow g_1^w$; Return (g_1, g_2, K)

Algorithm $\mathsf{Enc}((g_1, g_2, K), (e, f, h), M)$
$u \xleftarrow{\$} \mathbb{Z}_p$ $\boxed{^*}$
$a_1 \leftarrow g_1^u$; $a_2 \leftarrow g_2^u$; $b \leftarrow h^u$
$c \leftarrow b \cdot M$; $v \leftarrow H(K, (a_1, a_2, c))$
$d \leftarrow e^u f^{uv}$; Return (a_1, a_2, c, d)

Algorithm $\mathsf{KG}(g_1, g_2, K)$
$x_1, x_2, y_1, y_2, z_1, z_2 \xleftarrow{\$} \mathbb{Z}_p$
$e \leftarrow g_1^{x_1} g_2^{x_2}$; $f \leftarrow g_1^{y_1} g_2^{y_2}$; $h \leftarrow g_1^{z_1} g_2^{z_2}$
Return $((e, f, h), (x_1, x_2, y_1, y_2, z_1, z_2))$

Algorithm $\mathsf{Dec}((g_1, g_2, K), (e, f, h), (x_1, x_2, y_1, y_2, z_1, z_2), C)$
$(a_1, a_2, c, d) \leftarrow C$; $v \leftarrow H(K, (a_1, a_2, c))$; $M \leftarrow c \cdot a_1^{-z_1} a_2^{-z_2}$
If $d \neq a_1^{x_1 + y_1 v} a_2^{x_2 + y_2 v}$ Then $M \leftarrow \perp$
$\boxed{\text{If } a_1 = \mathbf{1} \text{ Then } M \leftarrow \perp}$
Return M

Figure G.9: The original CS scheme [CS03] does not contain the boxed code while the variant $\mathcal{CS}^*$ does. Although not shown above, the decryption algorithm in both versions always checks to ensure that the ciphertext $C \in \mathbb{G}^4$. The message space is $\mathbb{G}$.

that it is easy, given *cpars*, x, *dec*, to find *com* so that $\mathsf{Ver}(cpars, x, com, dec) = 1$. (This is true for any commitment scheme where *dec* is the coins used by the Com algorithm.) Consider then the AI-ATK adversary A against the transformed scheme that that receives a challenge ciphertext (C^*, com^*) where $C^* \leftarrow \mathsf{Enc}(pars, \mathsf{EK}[id_b], M^* \| dec^*)$ for hidden bit $b \in \{0, 1\}$. It then creates a commitment $\hat{com}$ of $\mathsf{EK}[id_1]$ with opening information $\hat{dec}$, and queries $(C^*, \hat{com})$ to be decrypted under $\mathsf{DK}[id_0]$. If $b = 0$ this query will probably return $\perp$ because $\mathsf{Ver}(cpars, \mathsf{EK}[id_0], \hat{com}, dec^*)$ is unlikely to be 1, but if $b = 1$ it returns $\hat{M}$, allowing A to determine the value of b. The weak robustness of $\mathcal{GE}$ rules out such anomalies.

G.5 A SROB-CCA version of Cramer-Shoup

Let $\mathbb{G}$ be a group of prime order p, and H: $\mathsf{Keys}(H) \times \mathbb{G}^3 \to \mathbb{G}$ a family of functions. We assume $\mathbb{G}, p, H$ are fixed and known to all parties. Figure G.9 shows the Cramer-Shoup (CS) scheme and the variant $\mathcal{CS}^*$ scheme where $\mathbf{1}$ denotes the identity element of $\mathbb{G}$. The differences are boxed. Recall that the CS scheme was shown to be IND-CCA in [CS03] and ANO-CCA in [BBDP01]. However, for any message $M \in \mathbb{G}$ the ciphertext $(\mathbf{1}, \mathbf{1}, M, \mathbf{1})$ in the CS scheme decrypts to M under *any pars*, *pk*, and *sk*, meaning in particular that the scheme is not even SROB-CPA. The modified scheme $\mathcal{CS}^*$ —which continues to be IND-CCA and ANO-CCA— removes this pathological case by having Enc choose the randomness u to be non-zero —Enc draws u from $\mathbb{Z}_p^*$ while the CS scheme draws it from $\mathbb{Z}_p$— and then having Dec reject (a_1, a_2, c, d) if $a_1 = \mathbf{1}$. This thwarts the attack, but is there any other attack? We show that there is not by proving that $\mathcal{CS}^*$ is actually SROB-CCA. Our proof of robustness relies only on the security —specifically, pre-image resistance— of the hash family H: it does not make the DDH assumption. Our proof uses ideas from the information-theoretic part of the proof of [CS03].

We say that a family H: $\mathsf{Keys}(H) \times \mathsf{Dom}(H) \to \mathsf{Rng}(H)$ of functions is *pre-image resistant* if, given a key K and a *random* range element v^*, it is computationally infeasible to find a pre-image of v^* under $H(K, \cdot)$. The notion is captured formally by the following advantage measure for an adversary I:

$$\mathbf{Adv}_H^{\text{pre-img}}(I) = \Pr\left[H(K, x) = v^* \ : \ K \xleftarrow{\$} \mathsf{Keys}(H) \ ; \ v^* \xleftarrow{\$} \mathsf{Rng}(H) \ ; \ x \xleftarrow{\$} I(K, v^*) \right] .$$

Pre-image resistance is not implied by the standard notion of one-wayness, since in the latter the target v^* is the image under $H(K, \cdot)$ of a random domain point, which may not be a random range point. However, it seems like a fairly mild assumption on a practical cryptographic hash function and is implied by the notion of "everywhere pre-image resistance" of [RS04], the difference being that, for the latter, the advantage is the maximum probability over all $v^* \in \mathsf{Rng}(H)$. We now

claim the following.

Theorem G.5.1 Let B be an adversary making two **GetEK** queries, no **GetDK** queries and at most $q-1$ **Dec** queries, and having running time t. Then we can construct an adversary I such that

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathcal{CS}^*}(A) \;\le\; \mathbf{Adv}^{\text{pre-img}}_{H}(I) + \frac{2q+1}{p}\,. \tag{G.1}$$

Furthermore, the running time of I is $t + q \cdot O(t_{\mathrm{exp}})$ where t_{exp} denotes the time for one exponentiation in $\mathbb{G}$.

Since $\mathcal{CS}^*$ is a PKE scheme, the above automatically implies security even in the presence of multiple **GetEK** and **GetDK** queries as required by game $\mathrm{SROB}_{\mathcal{CS}^*}$. Thus the theorem implies that $\mathcal{CS}^*$ is SROB-CCA if H is pre-image resistant. A detailed proof of Theorem G.5.1 is in Appendix G.10. Here we sketch some intuition.

We begin by conveniently modifying the game interface. We replace B with an adversary A that gets input $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$ representing the parameters that would be input to B and the public keys returned in response to B's two **GetEK** queries. Let $(x_{01}, x_{02}, y_{01}, y_{02}, z_{01}, z_{02})$ and $(x_{11}, x_{12}, y_{11}, y_{12}, z_{11}, z_{12})$ be the corresponding secret keys. The decryption oracle takes (only) a ciphertext and returns its decryption under *both* secret keys, setting a Win flag if these are both non-$\perp$. Adversary A no longer needs an output, since it can win via a **Dec** query.

Suppose A makes a **Dec** query (a_1, a_2, c, d). Then the code of the decryption algorithm Dec from Figure G.9 tells us that, for this to be a winning query, it must be that

$$d \;=\; a_1^{x_{01}+y_{01}v} a_2^{x_{02}+y_{02}v} \;=\; a_1^{x_{11}+y_{11}v} a_2^{x_{12}+y_{12}v}$$

where $v = H(K, (a_1, a_2, c))$. Letting $u_1 = \log_{g_1}(a_1), u_2 = \log_{g_2}(a_2)$ and $s = \log_{g_1}(d)$, we have

$$s \;=\; u_1(x_{01} + y_{01}v) + wu_2(x_{02} + y_{02}v) \;=\; u_1(x_{11} + y_{11}v) + wu_2(x_{12} + y_{12}v) \tag{G.2}$$

However, even acknowledging that A knows little about $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) through its **Dec** queries, it is unclear why Equation (G.2) is prevented by pre-image resistance —or in fact any property short of being a random oracle— of the hash function H. In particular, there seems no way to "plant" a target v^* as the value v of Equation (G.2) since the adversary controls u_1 and u_2. However, suppose now that $a_2 = a_1^w$. (We will discuss later why we can assume this.) This implies $wu_2 = wu_1$ or $u_2 = u_1$ since $w \neq 0$. Now from Equation (G.2) we have

$$u_1(x_{01} + y_{01}v) + wu_1(x_{02} + y_{02}v) - u_1(x_{11} + y_{11}v) - wu_1(x_{12} + y_{12}v) \;=\; 0\,.$$

We now see the value of enforcing $a_1 \neq 1$, since this implies $u_1 \neq 0$. After canceling u_1 and re-arranging terms, we have

$$v(y_{01} + wy_{02} - y_{11} - wy_{12}) + (x_{01} + wx_{02} - x_{11} - wx_{12}) \;=\; 0\,. \tag{G.3}$$

Given that $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) and w are chosen by the game, there is at most one solution v (modulo p) to Equation (G.3). We would like now to design I so that on input K, v^* it chooses $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ ($b \in \{0,1\}$) so that the solution v to Equation (G.3) is v^*. Then (a_1, a_2, c) will be a pre-image of v^* which I can output.

To make all this work, we need to resolve two problems. The first is why we may assume $a_2 = a_1^w$ —which is what enables Equation (G.3)— given that a_1, a_2 are chosen by A. The second is to properly design I and show that it can simulate A correctly with high probability. To solve these problems, we consider, as in [CS03], a modified check under which decryption, rather than rejecting when $d \neq a_1^{x_1+y_1v} a_2^{x_2+y_2v}$, rejects when $a_2 \neq a_1^w$ or $d \neq a_1^{x+yv}$, where $x = x_1 + wx_2$, $y = y_1 + wy_2$, $v = H(K, (a_1, a_2, c))$ and (a_1, a_2, c, d) is the ciphertext being decrypted. In our

proc Initialize
$cpars \xleftarrow{\$} \mathsf{CPG}$; $b \xleftarrow{\$} \{0,1\}$; Return $cpars$
proc LR(x_0, x_1)
$(com, dec) \xleftarrow{\$} \mathsf{Com}(cpars, x_b)$; Return com
proc Finalize(b')
Return $(b' = b)$

proc Initialize
$cpars \xleftarrow{\$} \mathsf{CPG}$; Return $cpars$
proc Finalize$(com, x_0, dec_0, x_1, dec_1)$
$d_0 \leftarrow \mathsf{Ver}(cpars, x_0, com, dec_0)$
$d_1 \leftarrow \mathsf{Ver}(cpars, x_1, com, dec_1)$
Return $(x_0 \neq x_1 \wedge d_0 = 1 \wedge d_1 = 1)$

Figure G.10: Game $\mathrm{HIDE}_{\mathcal{CMT}}$ (left) captures the hiding property while Game $\mathrm{BIND}_{\mathcal{CMT}}$ (right) captures the binding property. The adversary may call **LR** only once.

proof in Appendix G.10, games G_0–G_2 move us towards this perspective. Then, we fork off two game chains. Games G_3–G_6 are used to show that the modified decryption rule increases the adversary's advantage by at most $2q/p$. Games G_7–G_{11} show how to embed a target value v^* into the components of the secret key without significantly affecting the ability to answer **Dec** queries. Based on the latter, we then construct I as shown in Appendix G.10.

Acknowledgments

First and third authors were supported in part by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II. First author was supported in part by the French ANR-07-SESU-008-01 PAMPA Project. Second author was supported in part by NSF grants CNS-0627779 and CCF-0915675. Third author was supported in part by a Postdoctoral Fellowship from the Research Foundation – Flanders (FWO – Vlaanderen) and by the European Community's Seventh Framework Programme project PrimeLife (grant agreement no. 216483).

We thank Chanathip Namprempre, who declined our invitation to be a co-author, for her participation and contributions in the early stage of this work.

G.6 Appendix: Hiding and blinding of commitment schemes

The advantage measures

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(A) = 2 \cdot \Pr\left[\mathrm{HIDE}^{A}_{\mathcal{CMT}} \Rightarrow \mathsf{true}\right] - 1 \quad \text{and} \quad \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(A) = \Pr\left[\mathrm{BIND}^{A}_{\mathcal{CMT}} \Rightarrow \mathsf{true}\right],$$

which refer to the games of Figure G.10, capture, respectively, the standard hiding and binding properties of a commitment scheme. We refer to the corresponding notions as HIDE and BIND.

G.7 Appendix: More results on robustness of specific transforms and schemes

The Boneh-Franklin IBE. Boneh and Franklin proposed the first truly practical provably secure IBE scheme in [BF01]. They also propose a variant that uses the FO transform to obtain provable IND-CCA security in the random oracle model under the bilinear Diffie-Hellman (BDH) assumption; we refer to it as the BF-IBE scheme here. A straightforward modification of the proof can be used to show that BF-IBE is also ANO-CCA in the random oracle model under the same assumption. We now give a proof sketch that BF-IBE is also (unconditionally) SROB-CCA in the random oracle model.

Let e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a non-degenerate bilinear map, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are multiplicative cyclic groups of prime order p [BF01]. Let g be a generator of $\mathbb{G}_1$. The master secret key of the BF-IBE scheme is an exponent $s \xleftarrow{\$} \mathbb{Z}_p^*$, the public parameters contain $S \leftarrow g^s$. For random oracles $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^k$, $H_3 : \{0,1\}^k \times \{0,1\}^\ell \to \mathbb{Z}_p^*$, and $H_4 : \{0,1\}^k \to \{0,1\}^\ell$, the encryption of a message M under identity id is a tuple

$$\left(g^r \ , \ x \oplus H_2(e(S, H_1(id))^r) \ , \ M \oplus H_4(x)\right) \ ,$$

where $x \xleftarrow{\$} \{0,1\}^k$ and $r \leftarrow H_3(x, M)$. To decrypt a ciphertext (C_1, C_2, C_3), the user with identity id and decryption key $usk = H_1(id)^s$ computes $x \leftarrow C_2 \oplus H_2(e(C_1, usk))$, $M \leftarrow C_3 \oplus H_4(x)$, and $r \leftarrow H_3(x, M)$. If $C_1 \neq g^r$ he rejects, otherwise he outputs M.

Let us now consider a SROB-CCA adversary A that even knows the master secret s (and therefore can derive all keys and decrypt all ciphertexts that it wants). Since H_1 maps into $\mathbb{G}_1^*$, all its outputs are of full order p. The probability that A finds two identities id_1 and id_2 such that $H_1(id) = H_1(id_2)$ is negligible. Since $S \in \mathbb{G}_1^*$ and the map is non-degenerate, we therefore have that $g_{id_1} = e(S, H_1(id_1))$ and $g_{id_2} = e(S, H_1(id_2))$ are different and of full order p. Since H_3 maps into $\mathbb{Z}_p^*$, we have that $r \neq 0$, and therefore that $g_{id_1}^r$ and $g_{id_2}^r$ are different. If the output of H_2 is large enough to prevent collisions from being found, that also means that $H_2(g_{id_1}^r)$ and $H_2(g_{id_2}^r)$ are different. Decryption under both identities therefore yields two different values $x_1 \neq x_2$, and possibly different messages M_1, M_2. In order for the ciphertext to be valid for both identities, we need that $r = H_3(x_1, M_1) = H_3(x_2, M_2)$, but the probability of this happening is again negligible in the random oracle model. As a result, it follows that the BF-IBE scheme is also SROB-CCA in the random oracle model.

The Boyen-Waters IBE. Boyen and Waters [BW06] proposed a HIBE scheme which is IND-CPA and ANO-CPA in the standard model, and a variant that uses the CHK transform to achieve IND-CCA and ANO-CCA security. Decryption in the IND-CPA secure scheme never rejects, so it is definitely not SROB-CPA. Without going into details here, it is easy to see that the IND-CCA variant is not SROB-CPA either, because any ciphertext that is valid with respect to one identity will also be valid with respect to another identity, since the verification of the one-time signature does not depend on the identity of the recipient. (The natural fix to include the identity in the signed data may ruin anonymity.)

The IND-CCA-secure variant of Gentry's IBE scheme [Gen06] falls to a similar robustness attack as the original Cramer-Shoup scheme, by choosing a random exponent $r = 0$. We did not check whether explicitly forbidding this choice restores robustness, however.

G.8 Appendix: Application to auctions

Robustness of ElGamal. The parameters of the ElGamal encryption scheme consist of the description of a group $\mathbb{G}$ of prime order p with generator g. The secret key of a user is $x \xleftarrow{\$} \mathbb{Z}_p$, the corresponding public key is $X = g^x$. The encryption of a message M is the pair $(g^r, X^r \cdot M)$ for $r \xleftarrow{\$} \mathbb{Z}_p$. A ciphertext (R, S) is decrypted as $M \leftarrow R/S^x$. Since the decryption algorithm never returns $\perp$, the ElGamal scheme is obviously not robust. Stronger even, the ciphertext $(1, M)$ decrypts to M under any secret key. It is this strong failure of robustness that opens the way to attacks on applications like Sako's auction protocol [Sak00].

The protocol. Sako's auction protocol [Sak00] is important because it is the first truly practical one to hide the bids of losers. Let $1, \ldots, N$ be the range of possible bidding prices. In an initialization step, the auctioneer generates N ElGamal key pairs $(x_1, X_1), \ldots, (x_N, X_N)$, and publishes $g, X_1, \ldots, X_N$ and a fixed message $M \in \mathbb{G}$. A bidder places a bid of value

$v \in \{1, \ldots, N\}$ by encrypting M under X_v and posting the ciphertext. Note that the privacy of the bids is guaranteed by the anonymity of ElGamal encryption. The authority opens bids $C_1 = (R_1, S_1), \ldots, C_n = (R_n, S_n)$ by decrypting all bids under secret keys $x_N, \ldots, x_1$, until the highest index w where one or more bids decrypt to M. The auctioneer announces the identity of the winner(s), the price of the item w, and the secret key x_w. All auctioneers can then check that $S_i/R_i^{x_w} = M$ for all winners i.

An attack. Our attack permits a dishonest bidder and a colluding auctioneer to break the fairness of the protocol. (Security against colluding auctioneers was not considered in [Sak00], so we do not disprove their results, but it is a property that one may expect the protocol to have.) Namely, a cheating bidder can place a bid $(1, M)$. If w is the highest honest bid, then the auctioneer can agree to open the corrupted bid to with x_{w+1}, thereby winning the auction for the cheating bidder at one dollar more than the second-highest bidder.

Sako came close to preventing this attack with an "incompatible encryption" property that avoids choosing $r = 0$ at encryption. A dishonest bidder however may deviate from this encryption rule; the problem is that the decryption algorithm does not reject ciphertexts (R, S) when $R = 1$. The attack is easily prevented by using any of our robust encryption schemes, so that decryption under any other secret key than the intended one results in $\perp$ being returned. Note that for this application we really need the strong robustness notion with adversarially generated ciphertexts.

It is worth noting that, to enforce that all bids are independent of each other even in the presence of a colluding auctioneer, all bidders would also need to commit to their sealed bids (using a non-malleable commitment scheme) during a first round of communication and only open their commitments once all commitments made public.

G.9 Appendix: Proofs of Theorems G.4.1 and G.4.2

The proof of Part 2 of Theorem G.4.1 relies on the following information-theoretic lemma.

Lemma G.9.1 Let $\ell \leq k$ be positive integers and let A_1, A_2 be arbitrary algorithms with the length of the output of A_1 always being ℓ. Let P denote the probability that $A_2(A_1(K)) = K$ where the probability is over K drawn at random from $\{0,1\}^k$ and the coins of A_1, A_2. Then $P \leq 2^{\ell-k}$.

Proof of Lemma G.9.1: We may assume A_1, A_2 are deterministic for, if not, we can hardwire a "best" choice of coins for each. For each ℓ-bit string L let $S_L = \{K \in \{0,1\}^k : A_1(K) = L\}$ and let $s(L) = |S_L|$. Let $\mathcal{L}$ be the set of all $L \in \{0,1\}^\ell$ such that $s(L) > 0$. Then

$$\begin{aligned} P &= \sum_{L \in \mathcal{L}} \Pr[\, A_2(L) = K \mid A_1(K) = L \,] \cdot \Pr[\, A_1(K) = L \,] \\ &= \sum_{L \in \mathcal{L}} \frac{1}{s(L)} \cdot \frac{s(L)}{2^k} \\ &= \sum_{L \in \mathcal{L}} \frac{1}{2^k} \end{aligned}$$

which is at most $2^{\ell-k}$ as claimed. ▮

Proof of Part 2 of Theorem G.4.1: Games $\mathrm{G}_0, \mathrm{G}_1$ of Figure G.11 differ only in their **Finalize** procedures, with the message encrypted at line 04 to create ciphertext C in G_1 being

proc Initialize // $\mathrm{G}_0, \mathrm{G}_1$
01 $(pars, msk) \xleftarrow{\$} \mathsf{PG}$
02 $K \xleftarrow{\$} \{0,1\}^k$
03 $U, V \leftarrow \emptyset$
04 Return $(pars, K)$

proc GetEK(id) // $\mathrm{G}_0, \mathrm{G}_1$
01 $U \leftarrow U \cup \{id\}$
02 $(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
03 Return $\mathsf{EK}[id]$

proc GetDK(id) // $\mathrm{G}_0, \mathrm{G}_1$
01 If $id \notin U$ then return $\bot$
02 If $id \in \{id_0^*, id_1^*\}$ then return $\bot$
03 $V \leftarrow V \cup \{id\}$
04 Return $\mathsf{DK}[id]$

proc Dec(C, id) // $\mathrm{G}_0, \mathrm{G}_1$
01 If $id \notin U$ then return $\bot$
02 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
03 If $M = \bot$ then return $\bot$
04 $\overline{M} \| K^* \leftarrow M$
If $(K = K^*)$ then return $\overline{M}$
05 Else Return $\bot$

proc Finalize$(\overline{M}, id_0, id_1)$ // G_0
01 If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
02 If $(id_0 \in V) \vee (id_1 \in V)$ then return false
03 If $(id_0 = id_1)$ then return false
04 $\overline{M}_0 \leftarrow \overline{M}$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], \overline{M}_0 \| K)$
05 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
06 If $M = \bot$ then $\overline{M}_1 \leftarrow \bot$
07 Else
08 $\quad \overline{M}_1 \| K^* \leftarrow M$
09 $\quad$ If $(K \neq K^*)$ then $\overline{M}_1 \leftarrow \bot$
10 Return $(\overline{M}_0 \neq \bot) \wedge (\overline{M}_1 \neq \bot)$

proc Finalize$(\overline{M}, id_0, id_1)$ // G_1
01 If $(id_0 \notin U) \vee (id_1 \notin U)$ then return false
02 If $(id_0 \in V) \vee (id_1 \in V)$ then return false
03 If $(id_0 = id_1)$ then return false
04 $\overline{M}_0 \leftarrow \overline{M}$; $C \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_0], 0^{|\overline{M}_0|} \| 0^k)$
05 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id_1], \mathsf{DK}[id_1], C)$
06 If $M = \bot$ then $\overline{M}_1 \leftarrow \bot$
07 Else
08 $\quad \overline{M}_1 \| K^* \leftarrow M$
09 $\quad$ If $(K \neq K^*)$ then $\overline{M}_1 \leftarrow \bot$
10 Return $(\overline{M}_0 \neq \bot) \wedge (\overline{M}_1 \neq \bot)$

Figure G.11: Games for the proof of Part 2 of Theorem G.4.1.

a constant rather than $\overline{M}_0$ in G_0. We have

$$\mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(A) = \Pr\left[\mathrm{G}_0^A\right] = \left(\Pr\left[\mathrm{G}_0^A\right] - \Pr\left[\mathrm{G}_1^A\right]\right) + \Pr\left[\mathrm{G}_1^A\right] .$$

we design B so that

$$\Pr\left[\mathrm{G}_0^A\right] - \Pr\left[\mathrm{G}_1^A\right] \leq \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B) .$$

On input $pars$, adversary B executes lines 02,03 of **Initialize** and runs A on input $(pars, K)$. It replies to **GetEK**, **GetDK** and **Dec** queries of A via its own oracles of the same name. When A halts with output M, id_0, id_1, adversary B queries its **LR** oracle with $id_0, id_0, 0^{|M|} \| 0^k, M \| K$ to get back a ciphertext C. It then makes query **GetDK**(id_1) to get back $\mathsf{DK}[id_1]$. Note this is a legal query for B because id_1 is not one of the challenge identities in its **LR** query, but it would not have been legal for A. Now B executes lines 01–09 of the code of **Finalize** of G_1. If $\overline{M}_1 \neq \bot$ it outputs 1, else 0.

To complete the proof we show that $\Pr[\mathrm{G}_1^A] \leq 2^{\ell - k}$. We observe that M as computed at line 05 of **Finalize** in G_1 depends only on $pars, \mathsf{EK}[id_1], \mathsf{EK}[id_0], \mathsf{DK}[id_1], |\overline{M}_0|, k$. We would have liked to say that none of these depend on K. This would mean that the probability that $M \neq \bot$ and parses as $\overline{M}_1 \| K$ is at most 2^{-k}, making $\Pr[\mathrm{G}_1^A] \leq 2^{-k}$. In the PKE case, what we desire is almost true because the only item in our list that can depend on K is $|\overline{M}_0|$, which can carry at most $\log_2(t)$ bits of information about K. But id_0, id_1 could depend on K so in general, and in the IBE case in particular, $\mathsf{EK}[id_0], \mathsf{EK}[id_1], \mathsf{DK}[id_1]$ could depend on K. However we assumed that identities are n bits, so the total amount of information about K in the list $pars, \mathsf{EK}[id_1], \mathsf{EK}[id_0], \mathsf{DK}[id_1], |M_0|, k$ is at most $2n + \log_2(t)$ bits. We conclude by applying Lemma G.9.1 with $\ell = 2n + \lceil \log_2(t) \rceil$. ▮

Proof of Part 1 of Theorem G.4.2: Game G_0 of Figure G.12 is game $\mathrm{WROB}_{\overline{\mathcal{GE}}}$ tailored to the case that A makes only one **LR** query, an assumption we explained we can make. If we wish to exploit the assumed AI-ATK security of $\mathcal{GE}$, we need to be able to answer **Dec** queries of A using the **Dec** oracle in game $\mathrm{AI}_{\mathcal{GE}}$. Thus we would like to substitute the $\mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$ call in a $\mathbf{Dec}((C, com), id)$ query of G_0 with a $\mathbf{Dec}(C, id)$ call of an adversary B in $\mathrm{AI}_{\mathcal{GE}}$. The difficulty is that C might equal C^* but $com \neq com^*$, so that the call is not legal for B. To get around this, the first part of our proof will show that the decryption procedure of G_0 can be replaced by the alternative one of G_4, where this difficulty vanishes. This part exploits the uniqueness of the commitment scheme and the weak robustness of $\mathcal{GE}$. After that we will exploit the AI-ATK security of $\mathcal{GE}$ to remove dependence on dec^* in **LR**, allowing us to exploit the HIDE security of $\mathcal{CMT}$ to make the challenge commitment independent of $\mathsf{EK}[id_b^*]$. This allows us to conclude by again using the AI-ATK security of $\mathcal{GE}$. We proceed to the details.

In game G_0, if A makes a $\mathbf{Dec}((C^*, com), id_b^*)$ query with $com \neq com^*$ then the uniqueness of $\mathcal{CMT}$ implies that the procedure in question will return $\perp$. This means that line 02 of **Dec** in G_0 can be rewritten as line 02 of **Dec** in G_1 and the two procedures are equivalent. Procedure **Dec** of G_2 includes the boxed code and hence is equivalent to procedure **Dec** of G_1. Hence

$$\begin{aligned}
\frac{1}{2} + \frac{1}{2}\mathbf{Adv}^{\mathrm{ai}}_{\overline{\mathcal{GE}}}(A) = \Pr\left[\mathrm{G}_0^A\right] = \Pr\left[\mathrm{G}_1^A\right] &= \Pr\left[\mathrm{G}_2^A\right] \\
&= \Pr\left[\mathrm{G}_3^A\right] + \Pr\left[\mathrm{G}_2^A\right] - \Pr\left[\mathrm{G}_3^A\right] \\
&\leq \Pr\left[\mathrm{G}_3^A\right] + \Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right] .
\end{aligned}$$

The inequality above is by Lemma G.2.1 which applies because $\mathrm{G}_2, \mathrm{G}_3$ are identical until bad. We design W so that

$$\Pr\left[\mathrm{G}_3^A \text{ sets } \mathsf{bad}\right] \leq \mathbf{Adv}^{\mathrm{wrob}}_{\mathcal{GE}}(W) .$$

On input *pars*, adversary W executes lines 02,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to $\mathbf{GetEK}, \mathbf{GetDK}, \mathbf{Dec}$ queries of A via its own oracles of the same name, as per the code of G_3. When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary W executes lines 01,02,03 of the code of **LR** of G_3. It then outputs $\overline{M}_b^* \| dec^*, id_b^*, id_{1-b}^*$ and halts.

Next we bound $\Pr[\mathrm{G}_3^A]$. Procedure **Dec** of G_4 results from simplifying the code of procedure **Dec** of G_3, so

$$\Pr\left[\mathrm{G}_3^A\right] = \Pr\left[\mathrm{G}_4^A\right] = \left(\Pr\left[\mathrm{G}_4^A\right] - \Pr\left[\mathrm{G}_5^A\right]\right) + \Pr\left[\mathrm{G}_5^A\right] .$$

The step from G_4 to G_5 modifies only **LR**, replacing dec^* with a constant. We are assuming here that any decommitment key output by Com, regardless of the inputs to the latter, has length d bits. We design B_1 so that

$$\Pr\left[\mathrm{G}_4^A\right] - \Pr\left[\mathrm{G}_5^A\right] = \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B_1) .$$

On input *pars*, adversary B_1 executes lines 02,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to $\mathbf{GetEK}, \mathbf{GetDK}, \mathbf{Dec}$ queries of A via its own oracles of the same name, as per the code of G_4. Here we make crucial use of the fact that the alternative decryption rule of **Dec** of G_4 allows B_1 to respond to **Dec** queries of A without the need to query its own **Dec** oracle on (C^*, id_0^*) or (C^*, id_1^*). When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary B_1 executes lines 01,02,03 of the code of **LR** of G_4. It then queries $id_b^*, id_b^*, \overline{M}_b^* \| 0^d, \overline{M}_b^* \| dec^*$ to its own **LR** oracle to get back a ciphertext C^*, and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary B_1 outputs 1 if $b = b'$ and 0 otherwise.

proc Initialize // G_0–G_6
01 $(pars, msk) \xleftarrow{\$} \mathsf{PG}$
02 $cpars \xleftarrow{\$} \mathsf{CPG}$
03 $b \xleftarrow{\$} \{0,1\}$
04 $S, U, V \leftarrow \emptyset$; $C^* \leftarrow \bot$; $com^* \leftarrow \bot$
05 $id_0^* \leftarrow \bot$; $id_1^* \leftarrow \bot$
06 Return $(pars, cpars)$

proc GetEK(id) // G_0–G_6
01 $U \leftarrow U \cup \{id\}$
02 $(\mathsf{EK}[id], \mathsf{DK}[id]) \xleftarrow{\$} \mathsf{KG}(pars, msk, id)$
03 Return $\mathsf{EK}[id]$

proc GetDK(id) // G_0–G_6
01 If $id \notin U$ then return $\bot$
02 If $id \in \{id_0^*, id_1^*\}$ then return $\bot$
03 $V \leftarrow V \cup \{id\}$
04 Return $\mathsf{DK}[id]$

proc Finalize(b') // G_0–G_6
01 Return $(b' = b)$

proc LR$(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)$ // G_0–G_4
01 If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\bot$
02 If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\bot$
03 $(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, \mathsf{EK}[id_b^*])$
04 $C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| dec^*)$
05 Return (C^*, com^*)

proc LR$(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)$ // G_5
01 If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\bot$
02 If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\bot$
03 $(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, \mathsf{EK}[id_b^*])$
04 $C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| 0^d)$
05 Return (C^*, com^*)

proc LR$(id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*)$ // G_6
01 If $(id_0^* \notin U) \vee (id_1^* \notin U)$ then return $\bot$
02 If $(id_0^* \in V) \vee (id_1^* \in V)$ then return $\bot$
03 $(com^*, dec^*) \xleftarrow{\$} \mathsf{Com}(cpars, 0^e)$
04 $C^* \xleftarrow{\$} \mathsf{Enc}(pars, \mathsf{EK}[id_b^*], \overline{M}_b^* \| 0^d)$
05 Return (C^*, com^*)

proc Dec$((C, com), id)$ // G_0
01 If $id \notin U$ then return $\bot$
02 If $(id = id_b^*) \wedge (C, com) = (C^*, com^*)$ then return $\bot$
03 If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
04 Return $\bot$
05 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
06 If $M = \bot$ then return $\bot$
07 $\overline{M} \| dec \leftarrow M$
08 If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
09 Else return $\bot$

proc Dec$((C, com), id)$ // G_1
01 If $id \notin U$ then return $\bot$
02 If $(id = id_b^*) \wedge (C = C^*)$ then return $\bot$
03 If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
04 Return $\bot$
05 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
06 If $M = \bot$ then return $\bot$
07 $\overline{M} \| dec \leftarrow M$
08 If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
09 Else return $\bot$

proc Dec$((C, com), id)$ // $\boxed{G_2}$,G_3
01 If $id \notin U$ then return $\bot$
02 If $(id = id_b^*) \wedge (C = C^*)$ then return $\bot$
03 If $(id = id_{1-b}^* \neq id_b^*) \wedge (C, com) = (C^*, com^*)$ then
04 Return $\bot$
05 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
06 If $(id = id_{1-b}^* \neq id_b^*) \wedge (C = C^*) \wedge (com \neq com^*)$ then
07 $M^* \leftarrow M$
08 If $M \neq \bot$ then $\mathsf{bad} \leftarrow \mathsf{true}$; $M \leftarrow \bot$; $\boxed{M \leftarrow M^*}$
09 If $M = \bot$ then return $\bot$
10 $\overline{M} \| dec \leftarrow M$
11 If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
12 Else return $\bot$

proc Dec$((C, com), id)$ // G_4–G_6
01 If $id \notin U$ then return $\bot$
02 If $(id = id_0^*) \wedge (C = C^*)$ then return $\bot$
03 If $(id = id_1^*) \wedge (C = C^*)$ then return $\bot$
04 $M \leftarrow \mathsf{Dec}(pars, \mathsf{EK}[id], \mathsf{DK}[id], C)$
05 If $M = \bot$ then return $\bot$
06 $\overline{M} \| dec \leftarrow M$
07 If $\mathsf{Ver}(cpars, \mathsf{EK}[id], com, dec) = 1$ then return $\overline{M}$
08 Else return $\bot$

Figure G.12: Games for the proof of Part 1 of Theorem G.4.2.

Next we bound $\Pr[\mathrm{G}_5^A]$. Procedure **LR** of G_6 uses a constant 0^e rather than $\mathsf{EK}[id_b^*]$ as data for Com at line 03. The value of e is arbitrary, and we can just let $e = 1$. Then

$$\Pr\left[\,\mathrm{G}_5^A\,\right] = \left(\Pr\left[\,\mathrm{G}_5^A\,\right] - \Pr\left[\,\mathrm{G}_6^A\,\right]\right) + \Pr\left[\,\mathrm{G}_6^A\,\right] .$$

We design H so that

$$\Pr\left[\,\mathrm{G}_5^A\,\right] - \Pr\left[\,\mathrm{G}_6^A\,\right] \leq \mathbf{Adv}^{\mathrm{hide}}_{\mathcal{CMT}}(H) .$$

On input *cpars*, adversary H executes lines 01,03,04,05 of **Initialize** and runs A on input $(pars, cpars)$. It replies to $\mathbf{GetEK}, \mathbf{GetDK}, \mathbf{Dec}$ queries of A by direct execution of the code of these procedures in G_5, possible since it knows msk. When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary H executes lines 01,02 of the code of **LR** of G_5. It then queries $0^e, \mathsf{EK}[id_b^*]$ to its own **LR** oracle to get back a commitment com^*. It executes line 04 of **LR** of G_5 and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary H returns 1 if $b = b'$ and 0 otherwise.

Finally we design B_2 so that

$$2 \cdot \Pr\left[\,\mathrm{G}_6^A\,\right] - 1 \leq \mathbf{Adv}^{\mathrm{ai}}_{\mathcal{GE}}(B_2) .$$

On input *pars*, adversary B_2 executes lines 02,04,05 of **Initialize** and runs A on input (*pars*, *cpars*). It replies to $\mathbf{GetEK}, \mathbf{GetDK}, \mathbf{Dec}$ queries of A via its own oracles of the same name, as per the code of G_6. Again we make crucial use of the fact that the alternative decryption rule of **Dec** of G_6 allows B_2 to respond to **Dec** queries of A without the need to query its own **Dec** oracle on (C^*, id_0^*) or (C^*, id_1^*). When A makes its **LR** query $id_0^*, id_1^*, \overline{M}_0^*, \overline{M}_1^*$, adversary B_2 executes lines 01,02,03 of the code of **LR** of G_6. It then queries $id_0^*, id_1^*, \overline{M}_0^* \| 0^d, \overline{M}_1^* \| dec^*$ to its own **LR** oracle to get back a ciphertext C^*, and returns (C^*, com^*) to A. When A halts with outut a bit b', adversary B_2 outputs b'.

Adversary B of the theorem statement runs B_1 with probability 2/3 and B_2 with probability 1/3. ▌

Proof of Part 2 of Theorem G.4.2: In the execution of A with game $\mathrm{SROB}_{\overline{\mathcal{GE}}}$ let $\textsc{coll}$ be the event that there exist distinct id_0, id_1 queried by A to its **GetEK** oracle such that the encryption keys returned in response are the same. Then

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{srob}}_{\overline{\mathcal{GE}}}(A) &= \Pr\left[\,\mathrm{SROB}^{A}_{\overline{\mathcal{GE}}} \wedge \textsc{coll}\,\right] + \Pr\left[\,\mathrm{SROB}^{A}_{\overline{\mathcal{GE}}} \wedge \overline{\textsc{coll}}\,\right] \\
&\leq \Pr\left[\,\textsc{coll}\,\right] + \Pr\left[\,\mathrm{SROB}^{A}_{\overline{\mathcal{GE}}} \wedge \overline{\textsc{coll}}\,\right] .
\end{aligned}$$

But

$$\Pr\left[\,\textsc{coll}\,\right] \leq \binom{q}{2} \cdot \mathbf{Coll}_{\mathcal{GE}}$$

and we can design B such that

$$\Pr\left[\,\mathrm{SROB}^{A}_{\overline{\mathcal{GE}}} \wedge \overline{\textsc{coll}}\,\right] \leq \mathbf{Adv}^{\mathrm{bind}}_{\mathcal{CMT}}(B) .$$

We omit the details. ▌

G.10 Appendix: Proof of Theorem G.5.1

The proof relies on Games G_0–G_{11} of Figures G.13–G.15 and the adversary I of Figure G.16. See Section G.5 for intuition.

We begin by transforming B into an adversary A such that

$$\mathbf{Adv}^{\mathrm{srob}}_{\mathcal{CS}^*}(B) \;\le\; \Pr\left[\,\mathrm{G}_0^A\,\right]. \tag{G.4}$$

On input $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$, adversary A runs B on input (g_1, g_2, K). Adversary A returns to B the public key (e_0, f_0, h_0) in response to B's first **GetEK** query id_0, and (e_1, f_1, h_1) in response to its second **GetEK** query id_1. When B makes a **Dec** query, which can be assumed to have the form $(a_1, a_2, c, d), id_b$ for some $b \in \{0, 1\}$, adversary A queries (a_1, a_2, c, d) to its own **Dec** oracle to get back (M_0, M_1) and returns M_b to B. When B halts, with output that can be assumed to have the form $((a_1, a_2, c, d), id_0, id_1)$, adversary A makes a final query (a_1, a_2, c, d) to its **Dec** oracle and also halts.

We assume that every **Dec** query (a_1, a_2, c, d) of A satisfies $a_1 \neq \mathbf{1}$. This is without loss of generality because the decryption algorithm rejects otherwise. This will be crucial below. Similarly, we assume $(a_1, a_2, c, d) \in \mathbb{G}^4$. We now proceed to the analysis.

Games $\mathrm{G}_1, \mathrm{G}_2$ start to move us to the alternative decryption rule. In G_1, if $a_2 = a_1^w$ and $d = a_1^{x_b + y_b v}$ then $d = a_1^{x_{b1} + y_{b1} v} a_2^{x_{b2} + y_{b2} v}$, so **Dec** in G_1 returns the correct decryption, like in G_0. If $a_2 \neq a_1^w$ or $d \neq a_1^{x_b + y_b v}$ then, if $d \neq a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$, then **Dec** in G_1 returns $\perp$, else it returns $c a_1^{-z_{b1}} a_2^{-z_{b2}}$, so again is correct either way. Thus,

$$\begin{aligned}
\Pr\left[\,\mathrm{G}_0^A\,\right] &= \Pr\left[\,\mathrm{G}_1^A\,\right] \\
&= \Pr\left[\,\mathrm{G}_2^A\,\right] + (\Pr\left[\,\mathrm{G}_1^A\,\right] - \Pr\left[\,\mathrm{G}_2^A\,\right]) \\
&\le \Pr\left[\,\mathrm{G}_2^A\,\right] + \Pr\left[\,\mathrm{G}_2^A \text{ sets } \mathsf{bad}\,\right],
\end{aligned} \tag{G.5}$$

where the last line is by Lemma G.2.1 since $\mathrm{G}_1, \mathrm{G}_2$ are identical until bad. We now fork off two game chains, one to bound each term above.

First, we will bound the second term in the right-hand side of Inequality (G.5). Our goal is to move the choices of $x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2}$ $(b = 0, 1)$ and the setting of bad into **Finalize** while still being able to answer **Dec** queries. We will then be able to bound the probability that bad is set by a static analysis. Consider Game G_3. If $a_2 \neq a_1^w$ and $d = a_1^{x_{b1} + y_{b1} v} a_2^{x_{b2} + y_{b2} v}$ then bad is set in G_2. But $a_2 = a_1^w$ and $d \neq a_1^{x_b + y_b v}$ implies $d \neq a_1^{x_{b1} + y_{b1} v} a_2^{x_{b2} + y_{b2} v}$, so bad is not set in G_2. So,

$$\Pr\left[\,\mathrm{G}_2^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_3^A \text{ sets } \mathsf{bad}\,\right]. \tag{G.6}$$

Since we are only interested in the probability that G_3 sets bad, we have it always return true. The flag bad may be set at line 315, but is not used, so we move the setting of bad into the **Finalize** procedure in G_4. This requires that G_4 do some bookkeeping. We have also done some restructuring, moving some loop invariants out of the loop in **Dec**. We have

$$\Pr\left[\,\mathrm{G}_3^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_4^A \text{ sets } \mathsf{bad}\,\right]. \tag{G.7}$$

The choice of x_{b1}, x_{b2}, x_b at lines 404, 405 can equivalently be written as first choosing x_b and x_{b2} at random and then setting $x_{b1} = x_b - w x_{b2}$. This is true because w is not equal to 0 modulo p. The same is true for y_{b1}, y_{b2}, y_b. Once this is done, $x_{b1}, x_{b2}, y_{b1}, y_{b2}$ are not used until **Finalize**, so their choice can be delayed. Game G_5 makes these changes, so we have

$$\Pr\left[\,\mathrm{G}_4^A \text{ sets } \mathsf{bad}\,\right] \;=\; \Pr\left[\,\mathrm{G}_5^A \text{ sets } \mathsf{bad}\,\right]. \tag{G.8}$$

proc Initialize Game G_0

000 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
001 $K \xleftarrow{\$} \mathsf{Keys}(H)$
002 For $b = 0, 1$ do
003 $x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
004 $e_b \leftarrow g_1^{x_{b1}} g_2^{x_{b2}}$
005 $f_b \leftarrow g_1^{y_{b1}} g_2^{y_{b2}}$
006 $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
007 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Games G_1,G_2,G_3,G_4

100 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
101 $K \xleftarrow{\$} \mathsf{Keys}(H)$
102 For $b = 0, 1$ do
103 $x_{b1}, x_{b2}, y_{b1}, y_{b2}, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
104 $x_b \leftarrow x_{b1} + w x_{b2}$; $y_b \leftarrow y_{b_1} + w y_{b2}$
105 $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
106 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Finalize Games G_0,G_1,G_2

020 Return Win

proc Finalize Game G_3

320 Return true

proc Finalize Game G_4

420 For $b = 0, 1$ do
421 For all $(a_1, a_2, c, d, v) \in S$ do
422 If $d = a_1^{x_{b1}+y_{b1}v} \cdot a_2^{x_{b2}+y_{b2}v}$ Then
423 bad $\leftarrow$ true
424 Return true

proc Dec$((a_1, a_2, c, d))$ Game G_0

010 $v \leftarrow H(K, (a_1, a_2, c))$
011 For $b = 0, 1$ do
012 $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
013 If $d \neq a_1^{x_{b1}+y_{b1}v} \cdot a_2^{x_{b2}+y_{b2}v}$ Then $M_b \leftarrow \perp$
014 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then Win $\leftarrow$ true
015 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Games $\boxed{G_1}$,G_2

110 $v \leftarrow H(K, (a_1, a_2, c))$
111 For $b = 0, 1$ do
112 $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
113 If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b+y_b v})$ Then
114 $M_b \leftarrow \perp$
115 If $d = a_1^{x_{b1}+y_{b1}v} \cdot a_2^{x_{b2}+y_{b2}v}$ Then
116 bad $\leftarrow$ true ; $\boxed{M_b \leftarrow c a_1^{-z_{b1}} a_2^{-z_{b2}}}$
117 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then Win $\leftarrow$ true
118 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Game G_3

310 $v \leftarrow H(K, (a_1, a_2, c))$
311 For $b = 0, 1$ do
312 $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
313 If $(a_2 \neq a_1^w)$ Then
314 $M_b \leftarrow \perp$
315 If $d = a_1^{x_{b1}+y_{b1}v} \cdot a_2^{x_{b2}+y_{b2}v}$ Then bad $\leftarrow$ true
316 Return (M_0, M_1)

proc Dec$((a_1, a_2, c, d))$ Game G_4

410 $v \leftarrow H(K, (a_1, a_2, c))$
411 For $b = 0, 1$ do $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
412 If $(a_2 \neq a_1^w)$ Then
413 $S \leftarrow S \cup \{(a_1, a_2, c, d, v)\}$; $M_0, M_1 \leftarrow \perp$
414 Return (M_0, M_1)

Figure G.13: Games G_0, G_1, G_2, G_3, and G_4 for proof of Theorem G.5.1. G_1 includes the boxed code at line 116 but G_2 does not.

proc Initialize Games G_5,G_6

500 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
501 $K \xleftarrow{\$} \mathsf{Keys}(H)$; $S \leftarrow \emptyset$
502 For $b = 0, 1$ do
503 $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
504 $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
505 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Dec$((a_1, a_2, c, d))$ Games G_5,G_6

510 $v \leftarrow H(K, (a_1, a_2, c))$
511 For $b = 0, 1$ do $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
512 If $(a_2 \neq a_1^w)$ Then
513 $S \leftarrow S \cup \{(a_1, a_2, c, d, v)\}$; $M_0, M_1 \leftarrow \perp$
514 Return (M_0, M_1)

proc Finalize Game G_5

520 For $b = 0, 1$ do
521 $x_{b2}, y_{b2} \xleftarrow{\$} \mathbb{Z}_p$
522 $x_{b1} \leftarrow x_b - w x_{b2}$; $y_{b1} \leftarrow y_b - w y_{b2}$
523 For all $(a_1, a_2, c, d, v) \in S$ do
524 If $d = a_1^{x_{b1} + y_{b1} v} \cdot a_2^{x_{b2} + y_{b2} v}$ Then $\mathsf{bad} \leftarrow \mathsf{true}$
525 Return true

proc Finalize Game G_6

620 For $b = 0, 1$ do
621 $x_{b2}, y_{b2} \xleftarrow{\$} \mathbb{Z}_p$
622 $x_{b1} \leftarrow x_b - w x_{b2}$; $y_{b1} \leftarrow y_b - w y_{b2}$
623 For all $(a_1, a_2, c, d, v) \in S$ do
624 $u_1 \leftarrow \log_{g_1}(a_1)$; $u_2 \leftarrow \log_{g_2}(a_2)$
625 $s \leftarrow \log_{g_1}(d)$; $t_b \leftarrow s - u_1 x_b + u_1 y_b v$
626 $\alpha \leftarrow w(u_2 - u_1)$; $\beta \leftarrow wv(u_2 - u_1)$
627 If $t_b = \alpha x_{b2} + \beta y_{b2}$ Then $\mathsf{bad} \leftarrow \mathsf{true}$
628 Return true

Figure G.14: Games G_5 and G_6 for proof of Theorem G.5.1.

Game G_6 simply writes the test of line 524 in terms of the exponents. Note that this game computes discrete logarithms, but it is only used in the analysis and does not have to be efficient. We have

$$\Pr\left[\, G_5^A \text{ sets } \mathsf{bad} \,\right] \;=\; \Pr\left[\, G_6^A \text{ sets } \mathsf{bad} \,\right] . \tag{G.9}$$

We claim that

$$\Pr\left[\, G_6^A \text{ sets } \mathsf{bad} \,\right] \;\leq\; \frac{2q}{p} , \tag{G.10}$$

(Recall q is the number of **Dec** queries made by A.) We now justify Equation (G.10). By the time we reach **Finalize** in G_6, we can consider the adversary coins, all random choices of **Initialize**, and all random choices of **Dec** to be fixed. We will take probability only over the choice of x_{b2}, y_{b2} made at line 621. Consider a particular $(a_1, a_2, c, d, v) \in S$. This is now fixed, and so are the quantities $u_1, u_2, s, t_0, t_1, \alpha$ and β as computed at lines 624–626. So we want to bound the probability that bad is set at line 627 when we regard t_b, α, β as fixed and take the probability over the random choices of x_{b2}, y_{b2}. The crucial fact is that $u_2 \neq u_1$ because $(a_1, a_2, c, d, v) \in S$, and lines 612, 613 only put a tuple in S if $a_2 \neq a_1^w$. So α and β are not 0 modulo p, and the probability that $t_b = \alpha x_{b2} + \beta y_{b2}$ is thus $1/p$. The size of S is at most q so line 627 is executed at most $2q$ times. Equation (G.10) follows from the union bound.

We now return to Equation (G.5) to bound the first term. Game G_7 removes from G_2 code that does not affect outcome of the game. Once this is done, $x_{b1}, y_{b1}, x_{b2}, y_{b2}$ are used only to define x_b, y_b, so G_7 picks only the latter. So we have

$$\Pr\left[\, G_2^A \,\right] \;=\; \Pr\left[\, G_7^A \,\right] . \tag{G.11}$$

proc Initialize Game G_7

700 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$
701 $K \xleftarrow{\$} \mathsf{Keys}(H)$
702 For $b = 0, 1$ do
703 $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
704 $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
705 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Dec$((a_1, a_2, c, d))$ Games G_7–G_{11}

710 $v \leftarrow H(K, (a_1, a_2, c))$
711 For $b = 0, 1$ do
712 $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
713 If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b + y_b v})$ Then $M_b \leftarrow \perp$
714 If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $\textsc{Win} \leftarrow \mathsf{true}$
715 Return (M_0, M_1)

proc Finalize Games G_7–G_{11}

720 Return $\textsc{Win}$

proc Initialize Game G_8/$\boxed{G_9}$

800 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$
801 For $b = 0, 1$ do
802 $x_b, y_b, z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$
803 $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
804 If $y_1 = y_0$ Then
805 $\mathsf{bad} \leftarrow \mathsf{true}$; $\boxed{y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}}$
806 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Game G_{10}

1000 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$
1001 $x_0, y_0, x_1 \xleftarrow{\$} \mathbb{Z}_q$; $y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}$
1002 For $b = 0, 1$ do
1003 $z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$
1004 $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
1005 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

proc Initialize Game G_{11}

1100 $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $K \xleftarrow{\$} \mathsf{Keys}(H)$; $v^* \xleftarrow{\$} \mathbb{Z}_q$
1101 $x_0, y_0 \xleftarrow{\$} \mathbb{Z}_q$; $y_1 \xleftarrow{\$} \mathbb{Z}_q - \{y_0\}$; $x_1 \leftarrow x_0 - (y_1 - y_0)v^*$
1102 For $b = 0, 1$ do $z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
1103 Return $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$

Figure G.15: Games G_7–G_{11} for proof of Theorem G.5.1. G_9 includes the boxed code at line 805 but G_8 does not.

Game G_8 is the same as G_7 barring setting a flag that does not affect the game outcome, so

$$\begin{aligned}
\Pr\left[\mathrm{G}_7^A\right] &= \Pr\left[\mathrm{G}_8^A\right] \\
&= \Pr\left[\mathrm{G}_9^A\right] + \Pr\left[\mathrm{G}_8^A\right] - \Pr\left[\mathrm{G}_9^A\right] \\
&\leq \Pr\left[\mathrm{G}_9^A\right] + \Pr\left[\mathrm{G}_8^A \text{ sets } \mathsf{bad}\right] \qquad \text{(G.12)} \\
&\leq \Pr\left[\mathrm{G}_9^A\right] + \frac{1}{p}\,. \qquad \text{(G.13)}
\end{aligned}$$

Equation (G.12) is by Lemma G.2.1 since G_8, G_9 are identical until bad. The probability that G_8 sets bad is the probability that $y_1 = y_0$ at line 805, and this is $1/p$ since y is chosen at random from $\mathbb{Z}_p$, justifying Equation (G.13). The distribution of y_1 in G_9 is always uniform over $\mathbb{Z}_q - \{y_0\}$, and the setting of bad at line 805 does not affect the game outcome, so

$$\Pr\left[\mathrm{G}_9^A\right] = \Pr\left[\mathrm{G}_{10}^A\right]. \qquad \text{(G.14)}$$

Game G_{11} picks x_b, y_b differently from G_{10}, but since $y_1 - y_0 \neq 0$, the two ways induce the same distribution on x_0, x_1, y_0, y_1. Thus,

$$\Pr\left[\mathrm{G}_{10}^A\right] = \Pr\left[\mathrm{G}_{11}^A\right]. \qquad \text{(G.15)}$$

We now claim that

$$\Pr\left[\mathrm{G}_{11}^A\right] \leq \mathbf{Adv}_H^{\text{pre-img}}(I) \qquad \text{(G.16)}$$

where I is depicted in Figure G.16. To justify this, say that the A makes a **Dec** query (a_1, a_2, c, d)

Adversary $I(K, v^*)$
  $g_1 \xleftarrow{\$} \mathbb{G}^*$; $w \xleftarrow{\$} \mathbb{Z}_p^*$; $g_2 \leftarrow g_1^w$; $x_0, y_0 \xleftarrow{\$} \mathbb{Z}_p$; $y_1 \xleftarrow{\$} \mathbb{Z}_p - \{y_0\}$; $x_1 \leftarrow x_0 - (y_1 - y_0)v^*$
  For $b = 0, 1$ do
    $z_{b1}, z_{b2} \xleftarrow{\$} \mathbb{Z}_p$; $e_b \leftarrow g_1^{x_b}$; $f_b \leftarrow g_1^{y_b}$; $h_b \leftarrow g_1^{z_{b1}} g_2^{z_{b2}}$
  Run A on $(g_1, g_2, K), (e_0, f_0, h_0), (e_1, f_1, h_1)$
  On query $\mathbf{Dec}((a_1, a_2, c, d))$
    $v \leftarrow H(K, (a_1, a_2, c))$
    For $b = 0, 1$ do
      $M_b \leftarrow c \cdot a_1^{-z_{b1}} a_2^{-z_{b2}}$
      If $(a_2 \neq a_1^w \vee d \neq a_1^{x_b + y_b v})$ Then $M_b \leftarrow \perp$
      If $(M_0 \neq \perp) \wedge (M_1 \neq \perp)$ Then $(a_1^*, a_2^*, c^*) \leftarrow (a_1, a_2, c)$
    Return (M_0, M_1) to A
  Until A halts
  Return (a_1^*, a_2^*, c^*)

Figure G.16: Adversary I for proof of Theorem G.5.1.

which returns (M_0, M_1) with $M_0 \neq \perp$ and $M_1 \neq \perp$. This means we must have

$$d \;=\; a_1^{x_0 + y_0 v} \;=\; a_1^{x_1 + y_1 v}\;, \tag{G.17}$$

where $v = H(K, (a_1, a_2, c))$. Let $u_1 = \log_{g_1}(a_1)$ and $s = \log_{g_1}(d)$. Now, the above implies $u_1(x_0 + y_0 v) = u_1(x_1 + y_1 v)$. But (a_1, a_2, c, d) is a **Dec** query, and we know that $a_1 \neq \mathbf{1}$, so $u_1 \neq 0$. (This is a crucial point. Recall the reason we can without loss of generality assume $a_1 \neq \mathbf{1}$ is that the decryption algorithm of $\mathcal{CS}^*$ rejects otherwise.) Dividing u_1 out, we get $x_0 + y_0 v = x_1 + y_1 v$. Rearranging terms, we get $(y_1 - y_0)v = x_0 - x_1$. However, we know that $y_1 \neq y_0$, so $v = (y_1 - y_0)^{-1}(x_0 - x_1)$. However, this is exactly the value v^* due to the way I and Game G_{11} define x_0, y_0, x_1, y_1. Thus, we have $H(K, (a_1, a_2, c)) = v^*$, meaning I will be successful. Putting together Equations (G.4)–(G.11), (G.13)–(G.16) concludes the proof of Theorem G.5.1.

G.11 Appendix: Applications to searchable encryption

Public-key encryption with keyword search. A *public key encryption with keyword search* (PEKS) scheme [BDOP04] is a tuple $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ of algorithms. Via $(pk, sk) \xleftarrow{\$} \mathsf{KG}$, the key generation algorithm produces a pair of public and private keys. Via $C \xleftarrow{\$} \mathsf{PEKS}(pk, w)$, the encryption algorithm encrypts a keyword w to get a ciphertext under the public key pk. Via $t_w \xleftarrow{\$} \mathsf{Td}(sk, w)$, the trapdoor extraction algorithm computes a trapdoor t_w for keyword w. The deterministic test algorithm $\mathsf{Test}(t_w, C)$ returns 1 if C is an encryption of w and 0 otherwise. Privacy and consistency of PEKS schemes. We formulate privacy notions for PEKS using the games of Figure G.17. Let $\mathrm{ATK} \in \{\mathrm{CPA}, \mathrm{CCA}\}$. We define the advantage of an adversary A against the indistinguishability of $\mathcal{PEKS}$ as follows:

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathcal{PEKS}}(A) = \Pr\left[\,\text{IND-ATK}^{A}_{\mathcal{PEKS}} \Rightarrow \mathsf{true}\,\right].$$

We re-formulate the consistency definition of PEKS schemes of [ABC+08] using the game of Figure G.17. We define the advantage of an adversary A against the consistency of $\mathcal{PEKS}$ as

proc Initialize
$(pk, sk) \xleftarrow{\$} \mathsf{KG}$; $b \xleftarrow{\$} \{0,1\}$
$W \leftarrow \emptyset$; $C^* \leftarrow \perp$; Return pk

proc TD(w)
$\mathsf{TT}[w] \xleftarrow{\$} \mathsf{Td}(sk, w)$; $W \leftarrow W \cup \{w\}$; Return $\mathsf{TT}[w]$

proc LR(w_0^*, w_1^*)
$C^* \xleftarrow{\$} \mathsf{PEKS}(pk, w_b^*)$; Return C^*

proc Test(w, C)
If $(C = C^*) \wedge (w \in \{w_0^*, w_1^*\})$ Then return $\perp$
If $\mathsf{TT}[w] = \perp$ Then $\mathsf{TT}[w] \xleftarrow{\$} \mathsf{Td}(sk, w)$
Return $\mathsf{Test}(\mathsf{TT}[w], C)$

proc Finalize(b')
Return $(b = b') \wedge (\{w_0^*, w_1^*\} \cap W = \emptyset)$

proc Initialize
$(pk, sk) \xleftarrow{\$} \mathsf{KG}(pars)$
Return pk

proc Finalize(w, w')
$C \xleftarrow{\$} \mathsf{PEKS}(pk, w)$
$t' \xleftarrow{\$} \mathsf{Td}(sk, w')$
Return $(w \neq w') \wedge (\mathsf{Test}(t', C))$

Figure G.17: $\mathcal{PEKS} = (\mathsf{PG}, \mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$ is a PEKS scheme. Games IND-CCA$_{\mathcal{PEKS}}$ and IND-CPA$_{\mathcal{PEKS}}$ are on the left, where the latter omits procedure **Test**. The **LR** procedure may be called only once. Game CONSIST$_{\mathcal{PEKS}}$ is on the right.

follows:

$$\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A) = \Pr\left[\,\text{CONSIST}^{A}_{\mathcal{PEKS}} \Rightarrow \mathsf{true}\,\right].$$

Furthermore, we also recall the advantage measure $\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A)$, which captures the notion CONSIST of computational consistency of PEKS scheme $\mathcal{PEKS}$.

Transforming IBE to PEKS. The bdop-ibe-2-peks transform of [BDOP04] transforms an IBE scheme into a PEKS scheme. Given an IBE scheme $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$, the transform associates to it the PEKS scheme $\mathcal{PEKS} = (\mathsf{KG}, \mathsf{PEKS}, \mathsf{Td}, \mathsf{Test})$, where the key-generation algorithm KG returns $(pk, sk) \xleftarrow{\$} \mathsf{Setup}$; the encryption algorithm $\mathsf{PEKS}(pk, w)$ returns $C \leftarrow \mathsf{Enc}(pk, w, 0^k)$; the trapdoor extraction algorithm $\mathsf{Td}(sk, w)$ returns $t \xleftarrow{\$} \mathsf{Ext}(pk, sk, w)$; the test algorithm $\mathsf{Test}(t, C)$ returns 1 if and only if $\mathsf{Dec}(pk, t, C) = 0^k$. Abdalla et al. [ABC+08] showed that this transform generally does not provide consistency, and presented the consistency-providing new-ibe-2-peks transform as an alternative. We now show that the original bdop-ibe-2-peks transform does yield a consistent PEKS if the underlying IBE scheme is robust. We also show that if the base IBE scheme is ANO-CCA, then the PEKS scheme is IND-CCA, thereby yielding the first IND-CCA-secure PEKS schemes in the standard model, and the first consistent IND-CCA-secure PEKS schemes in the RO model. (Non-consistent IND-CCA-secure PEKS schemes in the RO model are easily derived from [FP07].)

Proposition G.11.1 Let $\mathcal{IBE}$ be an IBE scheme, and let $\mathcal{PEKS}$ be the PEKS scheme associated to it per the bdop-ibe-2-peks transform. Given any adversary A running in time t, we can construct an adversary B running in time $t + O(t)$ executions of the algorithms of $\mathcal{IBE}$ such that

$$\mathbf{Adv}^{\text{consist}}_{\mathcal{PEKS}}(A) \;\leq\; \mathbf{Adv}^{\text{srob-cpa}}_{\mathcal{IBE}}(B) \qquad \text{and} \qquad \mathbf{Adv}^{\text{ind-cca}}_{\mathcal{PEKS}}(A) \;\leq\; \mathbf{Adv}^{\text{ano-cca}}_{\mathcal{IBE}}(B)\,.$$

To see why the first inequality is true, it suffices to consider the adversary B that on input *pars* runs $(w, w') \xleftarrow{\$} A(pars)$ and outputs $C \xleftarrow{\$} \mathsf{Enc}(pars, w)$. The proof of the second inequality is an easy adaptation of the proof of the new-ibe-2-peks transform in [ABC+08], where B answers A's **Test** queries using its own **Dec** oracle.

Securely combining PKE and PEKS. Searchable encryption by itself is only of limited use since it can only encrypt individual keywords, and since it does not allow decryption. Fuhr and Paillier [FP07] introduce a more flexible variant that allows decryption of the keyword. An even more powerful (and general) primitive can be obtained by combining PEKS with PKE to encrypt non-searchable but recoverable content. For example, one could encrypt the body of an email using a PKE scheme, and append a list of PEKS-encrypted keywords. The straightforward approach of concatenating ciphertexts works fine for CPA security, but is insufficient for a strong, combined IND-CCA security model where the adversary has access to *both* a decryption oracle *and* a testing oracle. Earlier attempts to combine PKE and PEKS [BSNS06, ZI07] do not give the adversary access to the latter. A full IND-CCA-secure PKE/PEKS scheme in the standard model can be obtained by combining the IND-CCA-secure PEKS schemes obtained through our transformation with the techniques of [DK05]. Namely, one can consider label-based [Sho01] variants of the PKE and PEKS primitives, tie the different components of a ciphertext together by using as a common label the verification key of a one-time signature scheme, and append to the ciphertext a signature of all components under the corresponding signing key. We omit the details.

Bibliography

[ABC+05a] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, August 2005. (Cited on page 35, 36, 43, 44, 45.)

[ABC+05b] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. Cryptology ePrint Archive, 2005. `http://eprint.iacr.org/`. (Cited on page 195.)

[ABC+06] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, Bodo Möller, and David Pointcheval. Provably secure password-based authentication in TLS. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *ASIACCS 06: 1st Conference on Computer and Communications Security*, pages 35–45. ACM Press, March 2006. (Cited on page 22.)

[ABC+08] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, July 2008. (Cited on page 35, 36, 44, 45, 191, 226, 228, 230, 250, 251.)

[ABC+11] Michel Abdalla, James Birkett, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, Jacob C. N. Schuldt, and Nigel P. Smart. Wildcarded identity-based encryption. *Journal of Cryptology*, 24(1):42–82, January 2011. (Cited on page 40, 155.)

[ABCP06] Michel Abdalla, Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Password-based group key exchange in a constant number of rounds. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 427–442. Springer, April 2006. (Cited on page 28, 30, 117, 130.)

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, February 2010. (Cited on page 37, 225.)

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, April 2001. (Cited on page 63, 64, 227, 228.)

[ACCP08] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351. Springer, April 2008. (Cited on page 20, 28, 31.)

[ACD+06] Michel Abdalla, Dario Catalano, Alex Dent, John Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311. Springer, July 2006. (Cited on page 40, 41, 42, 157.)

[ACF09] Michel Abdalla, Dario Catalano, and Dario Fiore. Verifiable random functions from identity-based key encapsulation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 554–571. Springer, April 2009. (Cited on page 46.)

[ACGP11] Michel Abdalla, Céline Chevalier, Louis Granboulan, and David Pointcheval. Contributory password-authenticated group key exchange with join capability. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 142–160. Springer, February 2011. (Cited on page 31.)

[ACP05] Michel Abdalla, Olivier Chevassut, and David Pointcheval. One-time verifier-based encrypted key exchange. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 47–64. Springer, January 2005. (Cited on page 22.)

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, August 2009. (Cited on page 22, 26, 27, 79.)

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *29th Annual ACM Symposium on Theory of Computing*, pages 284–293. ACM Press, May 1997. (Cited on page 193.)

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 65–84. Springer, January 2005. (Cited on page 17, 118, 119, 120.)

[AFP06] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. *IEE Proceedings — Information Security*, 153(1):27–39, March 2006. (Cited on page 17.)

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 119–135. Springer, May 2001. (Cited on page 80.)

[AKN07] Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. In Joachim Biskup and Javier López, editors, *ESORICS 2007: 12th European Symposium on Research in Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 139–154. Springer, September 2007. (Cited on page 42.)

[AKN08] Michel Abdalla, Eike Kiltz, and Gregory Neven. Generalized key delegation for hierarchical identity-based encryption. *IET Information Security*, 2(3):67–78, September 2008. (Cited on page 42.)

[AP05] Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 191–208. Springer, February 2005. (Cited on page 21, 22, 59, 121.)

[AP06] Michel Abdalla and David Pointcheval. A scalable password-based group key exchange protocol in the standard model. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 332–347. Springer, December 2006. (Cited on page 22, 28, 29, 30, 32, 80, 82, 84, 129.)

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 2004. (Cited on page 33, 39, 40, 42, 160, 162, 165, 175, 180, 234.)

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, August 2004. (Cited on page 219.)

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, December 2001. (Cited on page 35, 194, 205, 208, 226, 227, 228, 230, 237.)

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer, May 2005. (Cited on page 41, 42, 164, 165, 170, 177, 180, 194, 209, 212, 219, 220.)

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, May 2000. (Cited on page 134.)

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188. Springer, May 2004. (Cited on page 22.)

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007. (Cited on page 228.)

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols (extended abstract). In *30th Annual ACM Symposium on Theory of Computing*, pages 419–428. ACM Press, May 1998. (Cited on page 16, 17.)

[BCL+05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377. Springer, August 2005. (Cited on page 22, 26, 80, 81, 89, 103.)

[BCP01] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably authenticated group Diffie-Hellman key exchange – the dynamic case. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 290–309. Springer, December 2001. (Cited on page 117, 118, 129.)

[BCP02a] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, April / May 2002. (Cited on page 117, 118.)

[BCP02b] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Group Diffie-Hellman key exchange secure against dictionary attacks. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 497–514. Springer, December 2002. (Cited on page 28, 117, 118, 119, 130.)

[BCP03] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security proofs for an efficient password-based key exchange. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 03: 10th Conference on Computer and Communications Security*, pages 241–250. ACM Press, October 2003. (Cited on page 21, 60, 62, 130.)

[BCP04] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. New security results on encrypted key exchange. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 145–158. Springer, March 2004. (Cited on page 21, 60, 62, 73, 74, 130.)

[BCP07] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. A security solution for IEEE 802.11's ad-hoc mode: Password authentication and group Diffie-Hellman

key exchange. *International Journal of Wireless and Mobile Computing*, 2(1):4–13, 2007. (Cited on page 28, 130, 131.)

[BCPQ01] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably authenticated group Diffie-Hellman key exchange. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 255–264. ACM Press, November 2001. (Cited on page 28, 117, 118, 129.)

[BD94] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system (extended abstract). In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 275–286. Springer, May 1994. (Cited on page 28, 29, 118, 120, 130, 131.)

[BD05] Mike Burmester and Yvo Desmedt. A secure and scalable group key exchange system. *Information Processing Letters*, 94(3):137–143, May 2005. (Cited on page 28, 29, 118, 120, 130, 131, 137.)

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science*, pages 394–403. IEEE Computer Society Press, October 1997. (Cited on page 82, 96, 98.)

[BDNS07] James Birkett, Alexander W. Dent, Gregory Neven, and Jacob C. N. Schuldt. Efficient chosen-ciphertext secure identity-based encryption with wildcards. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 274–292. Springer, July 2007. (Cited on page 157.)

[BDOP04] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, May 2004. (Cited on page 36, 43, 44, 45, 191, 192, 193, 194, 195, 196, 199, 200, 206, 226, 250, 251.)

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer, August 1998. (Cited on page 226, 230, 231.)

[Bel11] Mihir Bellare. CSE 207 – modern cryptography. Lecture Notes, 2011. `http://www-cse.ucsd.edu/users/mihir/cse207`. (Cited on page 3.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, August 2001. (Cited on page 9, 228, 239, 240.)

[BF03] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. (Cited on page 6, 33, 34, 36, 38, 39, 155, 161, 193, 194, 195, 197, 200, 205, 209, 212, 226, 230.)

[BFMLS08] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel P. Smart. Generic constructions of identity-based and certificateless KEMs. *Journal of Cryptology*, 21(2):178–199, April 2008. (Cited on page 165, 167, 173.)

[BG85] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 289–302. Springer, August 1985. (Cited on page 157.)

[BGH07] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th Annual Symposium on Foundations of Computer Science*, pages 647–657. IEEE Computer Society Press, October 2007. (Cited on page 230.)

[BGS06] Jens-Matthias Bohli, Maria Isabel Gonzalez Vasco, and Rainer Steinwandt. Password-authenticated constant-round group key establishment with a common reference string. Cryptology ePrint Archive, Report 2006/214, 2006. `http://eprint.iacr.org/`. (Cited on page 22, 80, 131.)

[BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111, 2011. (Cited on page 48.)

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 455–470. Springer, December 2008. (Cited on page 170.)

[BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, May 2003. (Cited on page 61.)

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Cited on page 15, 21, 60, 67, 73, 81, 82, 130.)

[BM93] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 244–250. ACM Press, November 1993. (Cited on page 21.)

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, August 1999. (Cited on page 6, 214.)

[BMP00] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer, May 2000. (Cited on page 17, 21, 62, 82, 130.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, January 2003. (Cited on page 80, 81.)

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000. (Cited on page 16, 21, 62, 82, 118, 130, 131.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. (Cited on page 18, 21, 39, 81, 82, 130, 162, 166, 196.)

[BR94a] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1994. (Cited on page 16, 63, 130.)

[BR94b] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, May 1994. (Cited on page 118.)

[BR95] Mihir Bellare and Phillip Rogaway. Provably secure session key distribution: The three party case. In *27th Annual ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, May / June 1995. (Cited on page 63, 130.)

[BR00] Mihir Bellare and Phillip Rogaway. The AuthA protocol for password-based authenticated key exchange. Contributions to IEEE P1363, March 2000. (Cited on page 60.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, May / June 2006. (Cited on page 8, 229.)

[BSNS06] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006: 9th International Conference on Information Security*, volume 4176 of *Lecture Notes in Computer Science*, pages 217–232. Springer, August / September 2006. (Cited on page 252.)

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. (Cited on page 171.)

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *Advances in Cryptology*

– *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer, August 2006. (Cited on page 195, 212, 219, 228, 240.)

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, February 2007. (Cited on page 195.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001. (Cited on page 16, 19, 81, 89.)

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, August 2001. (Cited on page 81, 82, 84, 89, 90.)

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218. ACM Press, May 1998. (Cited on page 22.)

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer, May 2003. (Cited on page 35, 39, 214.)

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, May 2004. (Cited on page 167, 182, 183, 228.)

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005. (Cited on page 19, 23, 24, 25, 26, 80, 81, 82, 84, 92, 101, 102, 103, 106.)

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, April / May 2002. (Cited on page 19, 81.)

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, December 2001. (Cited on page 33, 155.)

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998. (Cited on page 79, 82, 97, 130, 131, 136.)

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002. (Cited on page 22, 23, 79, 80, 84, 86, 130, 131, 134.)

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. (Cited on page 130, 157, 165, 173, 227, 228, 237, 238.)

[CS05] Sanjit Chatterjee and Palash Sarkar. Trading time for space: Towards an efficient IBE scheme with short(er) public parameters in the standard model. In Dongho Won and Seungjoo Kim, editors, *ICISC 05: 8th International Conference on Information Security and Cryptology*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer, December 2005. (Cited on page 178.)

[DB06] Ratna Dutta and Rana Barua. Password-based encrypted group key agreement. *International Journal of Network Security*, 3(1):30–41, July 2006. `http://isrc.nchu.edu.tw/ijns`. (Cited on page 29, 117, 118, 121, 130.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Cited on page 226, 230.)

[Den03] Alexander W. Dent. A designer's guide to KEMs. In Kenneth G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer, December 2003. (Cited on page 182, 185, 186.)

[DF92] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures (extended abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 457–469. Springer, August 1992. (Cited on page 6.)

[DFK+03] Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 19–32. Springer, April 2003. (Cited on page 6.)

[DG03] Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 507–523. Springer, May 2003. (Cited on page 61, 62.)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page 4, 5, 129.)

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st Annual Symposium on Foundations of Computer Science*, pages 511–520. IEEE Computer Society Press, 2010. (Cited on page 47.)

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pages 188–209. Springer, February 2005. (Cited on page 82, 96, 98, 252.)

[DKOS01] Giovanni Di Crescenzo, Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Efficient and non-interactive non-malleable commitment. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 40–59. Springer, May 2001. (Cited on page 84.)

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer, April / May 2002. (Cited on page 6.)

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 581–596. Springer, August 2002. (Cited on page 81, 89.)

[DNR04] Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, May 2004. (Cited on page 193.)

[DP06] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, September 2006. (Cited on page 80.)

[EGM96] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996. (Cited on page 23, 25, 27.)

[ElG85a] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. (Cited on page 7, 11, 12.)

[ElG85b] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1985. (Cited on page 82, 97.)

[FK04] Ian T. Foster and Carl Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004. (Cited on page 117.)

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, August 1999. (Cited on page 228.)

[FP07] Thomas Fuhr and Pascal Paillier. Decryptable searchable encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *Provable Security, First International Conference, ProvSec 2007*, volume 4784 of *Lecture Notes in Computer Science*, pages 228–236, Wollongong, Australia, November 1–2, 2007. Springer. (Cited on page 251, 252.)

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer, May / June 2006. (Cited on page 195, 240.)

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115. IEEE Computer Society Press, October 2003. (Cited on page 22.)

[GL01] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 408–432. Springer, August 2001. `http://eprint.iacr.org/2000/057`. (Cited on page 22, 62, 82.)

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. (Cited on page 22, 23, 26, 29, 30, 62, 79, 80, 81, 82, 84, 87, 92, 93, 96, 97, 101, 103, 111, 130, 131, 134, 136, 137.)

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. (Cited on page 11, 82, 125, 226, 230.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 133.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987. (Cited on page 81.)

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. (Cited on page 81.)

[Goh03] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/`. (Cited on page 191, 195.)

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. (Cited on page 125, 193, 198.)

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98. ACM

Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 171.)

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, December 2002. (Cited on page 36, 39, 156, 161, 194, 209, 212, 218.)

[GSW04] Philippe Golle, Jessica Staddon, and Brent R. Waters. Secure conjunctive keyword search over encrypted data. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pages 31–45. Springer, June 2004. (Cited on page 191, 195.)

[Hal05] Shai Halevi. A sufficient condition for key-privacy. Cryptology ePrint Archive, Report 2005/005, 2005. `http://eprint.iacr.org/`. (Cited on page 194, 208.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 32, 133.)

[HK99] Shai Halevi and Hugo Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and System Security*, 2(3):230–268, August 1999. (Cited on page 62.)

[HL02] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, April / May 2002. (Cited on page 34, 39, 156, 161, 194, 209, 212.)

[HW08] Dennis Hofheinz and Enav Weinreb. Searchable encryption with decryption in the standard model. Cryptology ePrint Archive, Report 2008/423, 2008. `http://eprint.iacr.org/`. (Cited on page 228.)

[IEE04] IEEE draft standard P1363.2. Password-based public key cryptography. `http://grouper.ieee.org/groups/1363/passwdPK`, May 2004. Draft Version 15. (Cited on page 62.)

[Jab97] David P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *6th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 1997)*, pages 248–255, Cambridge, MA, USA, June 18–20, 1997. IEEE Computer Society. (Cited on page 21.)

[Jou04] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, September 2004. (Cited on page 9, 160.)

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, May 2005. (Cited on page 79, 80.)

[KG09] Eike Kiltz and David Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theoretical Computer Science*, 410(47-49):5093–5111, 2009. (Cited on page 157.)

[KI02] Kazukuni Kobara and Hideki Imai. Pretty-simple password-authenticated key-exchange under standard assumptions. *IEICE Transactions*, E85-A(10):2229–2237, October 2002. Also available at `http://eprint.iacr.org/2003/038/`. (Cited on page 60, 61, 62.)

[KLL04] Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee. Constant-round authenticated group key exchange for dynamic groups. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 245–259. Springer, December 2004. (Cited on page 117, 118, 121, 130.)

[KMTG05] Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05: 3rd International Conference on Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 1–16. Springer, June 2005. (Cited on page 141.)

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001. (Cited on page 22, 60, 62, 80, 82, 101, 130, 131, 137, 141.)

[KOY02] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Forward secrecy in password-only key exchange protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 29–44. Springer, September 2002. (Cited on page 137.)

[Kra03] Hugo Krawczyk. SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425. Springer, August 2003. (Cited on page 5, 59.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, April 2008. (Cited on page 228.)

[KY03] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 110–125. Springer, August 2003. (Cited on page 17, 117, 118, 120, 125, 127, 130, 132, 141, 144.)

[LHL04] Su-Mi Lee, Jung Yeon Hwang, and Dong Hoon Lee. Efficient password-based group key exchange. In Sokratis K. Katsikas, Javier Lopez, and Günther Pernul, editors, *TrustBus 2004: Trust and Privacy in Digital Business, 1st International Conference*, volume 3184 of *Lecture Notes in Computer Science*, pages 191–199. Springer, August / September 2004. (Cited on page 29, 118, 122.)

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 725–734. ACM Press, June 2011. (Cited on page 47.)

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, May / June 2006. (Cited on page 80.)

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, May 2010. (Cited on page 48.)

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 70–88. Springer, March 2011. (Cited on page 47.)

[Luc97] Stefan Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Workshop on Security Protocols*, École Normale Supérieure, 1997. (Cited on page 21.)

[Mac02] Philip D. MacKenzie. The PAK suite: Protocols for password-authenticated key exchange. Contributions to IEEE P1363.2, 2002. (Cited on page 21, 60.)

[MMM02] Tal Malkin, Daniele Micciancio, and Sara K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417. Springer, April / May 2002. (Cited on page 216.)

[MPS00] Philip D. MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-authenticated key exchange based on RSA. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 599–613. Springer, December 2000. (Cited on page 62.)

[MSJ02] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385–400. Springer, August 2002. (Cited on page 62.)

[MSK02] Shigeo Mitsunari, Ryuichi Saka, and Masao Kasahara. A new traitor tracing. *IEICE Transactions*, E85-A(2):481–484, February 2002. (Cited on page 160.)

[Nac07] David Naccache. Secure and *practical* identity-based encryption. *IET Information Security*, 1(2):59–64, 2007. (Cited on page 178.)

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, August 2002. (Cited on page 22.)

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. ACM-SIAM, January 2001. (Cited on page 80.)

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, May 1999. (Cited on page 79.)

[PKL04] Dong Jin Park, Kihyun Kim, and Pil Joong Lee. Public key encryption with conjunctive field keyword search. In Chae Hoon Lim and Moti Yung, editors, *WISA 04: 5th International Workshop on Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, pages 73–86. Springer, August 2004. (Cited on page 195.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. (Cited on page 74.)

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, August 1992. (Cited on page 226, 230.)

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, February 2004. (Cited on page 237.)

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999. (Cited on page 25.)

[Sak00] Kazue Sako. An auction protocol which hides bids of losers. In Hideki Imai and Yuliang Zheng, editors, *PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 422–432. Springer, January 2000. (Cited on page 226, 240, 241.)

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949. (Cited on page 3.)

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, August 1985. (Cited on page 33, 155, 205.)

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, May 1997. (Cited on page 77.)

[Sho99] Victor Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. (Cited on page 17.)

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/`. (Cited on page 252.)

[Sho04] Victor Shoup. ISO 18033-2: An emerging standard for public-key encryption. `http://shoup.net/iso/std6.pdf`, December 2004. Final Committee Draft. (Cited on page 23, 25, 32, 82, 133.)

[Sma03] Nigel P. Smart. Access control using pairing based cryptography. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 111–121. Springer, April 2003. (Cited on page 171.)

[SNS88] Jennifer G. Steiner, B. Clifford Neuman, and Jeffrey L. Schiller. Kerberos: An authentication service for open networks. In *Proceedings of the USENIX Winter Conference*, pages 191–202, Dallas, TX, USA, 1988. (Cited on page 5.)

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *SCIS 2000*, Okinawa, Japan, January 2000. (Cited on page 155.)

[STW95] Michael Steiner, Gene Tsudik, and Michael Waidner. Refinement and extension of encrypted key exchange. *ACM SIGOPS Operating Systems Review*, 29(3):22–30, July 1995. (Cited on page 21.)

[STW00] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, August 2000. (Cited on page 131.)

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, May 2005. (Cited on page 170, 171.)

[SWP00] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pages 44–55. IEEE Computer Society Press, May 2000. (Cited on page 191, 195.)

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005. (Cited on page 41, 42, 165, 177, 194, 219.)

[WBDS04] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and Diana K. Smetters. Building an encrypted and searchable audit log. In *ISOC Network and Distributed System Security Symposium – NDSS 2004*. The Internet Society, February 2004. (Cited on page 191, 192, 195.)

[ZI07] Rui Zhang and Hideki Imai. Generic combination of public key encryption with keyword search and public key encryption. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *CANS 07: 6th International Conference on Cryptology and Network Security*, volume 4856 of *Lecture Notes in Computer Science*, pages 159–174. Springer, December 2007. (Cited on page 252.)

Abstract

Trusted parties are fundamental for the establishment of secure communication among users. Such is the case, for example, when establishing a trusted relationship between users and certain public information in a public-key infrastructure for public-key encryption and signature schemes or when storing high-entropy secret keys in a cryptographic device. Clearly, if the trusted party misbehaves in either of these situations, then the overall security of the scheme or protocol in which we are interested can be adversely affected.

There are several ways in which one can try to reduce the amount of trust in third parties, such as making the task of recovering the secret key harder for the adversary, as in distributed cryptosystems or minimizing the damage caused by secret-key exposures, as in forward-secure and intrusion-resilient cryptosystems. In this thesis, we consider two additional methods.

The first one, which goes by the name of password-based key exchange, is to assume that the secret keys used in authenticated key exchange protocols have low entropy and do not need to be stored in a cryptographic device. In spite of the low entropy of secret keys, such protocols can still provide a level of assurance which may be sufficient for most applications.

The second method for reducing the amount of trust in third parties is to use an identity-based cryptosystem, in which the public key of a user can be an arbitrary string such as an email address. As identity-based cryptosystems provide collusion resistance, they can also be used to lessen the damage caused by secret-key exposures by generating independent secret keys for different time periods or devices. Moreover, identity-based cryptosystems can allow users to have a more fine-grained control over the decryption capabilities of third parties, further limiting the harmful consequences due to their misbehavior.

Résumé

Les tiers de confiance sont essentiels aux communications sécurisées. Par exemple, dans une infrastructure de gestion de clés, l'autorité de certification est la clé de voute de l'authentification en garantissant le lien entre une identité et une clé publique. Une carte à puce se doit, pour sa part, d'assurer la confidentialité et l'intégrité des données secrètes lorsqu'elle sert de stockage de données cryptographiques. En effet, si ces garanties sont mises en défaut dans l'une de ces situations, alors la sécurité globale du système peut en être affectée.

Plusieurs approches permettent de réduire l'importance des tiers de confiance, telles qu'accroître la difficulté de recouvrer la clé secrète, en la distribuant parmi plusieurs entités, ou limiter l'impact d'une fuite d'information secrète, comme dans les cryptosystèmes « intrusion-resilient » ou « forward-secure ». Dans cette thèse, nous considérons deux méthodes complémentaires.

La première méthode consiste à utiliser des mots de passe, ou des clés secrètes de faible entropie, qui n'ont pas besoin d'être stockés dans un dispositif cryptographique sécurisé. Malgré la faible entropie du secret, de tels protocoles peuvent fournir un niveau d'assurance satisfaisant pour la plupart des applications. On considère en particulier la mise en accord de clés.

La deuxième méthode limite le besoin de garantie de la part des tiers de confiance en utilisant un cryptosystème basé sur l'identité, dans lequel la clé publique d'un utilisateur peut être une chaîne de caractères arbitraire, telle qu'une adresse email. Comme ces systèmes fournissent une résistance aux collusions, ils peuvent aussi être utilisés pour réduire les dommages causés par l'exposition de clés secrètes en générant des secrets indépendants pour chaque période de temps ou pour chaque périphérique/entité. Par ailleurs, ces systèmes basés sur l'identité permettent aux utilisateurs d'avoir un contrôle plus fin sur les capacités de déchiffrement des tiers, en limitant les conséquences liées à un mauvais usage.