Cryptographie de l'identité

Paul Germouty

Université de Limoges
Ecole Doctorale SISMI (610)
XLim

Thesis to Obtain the title of:
Docteur de l'Université de Limoges in Computer Science
Presentation and Defence by:

Paul Germouty

September 13, 2018

Identity-based Cryptography

Directed by: **Olivier Blazy** and **Duong Hieu Phan**

Reviewers:

Sébastien Canard *(Orange)*
Damien Vergnaud *(Sorbonne Université and Institut Universitaire de France)*

Examinators:

Céline Chevalier *(Université Paris II - Panthéon Assas)*
Philippe Gaborit *(Université de Limoges)*

Université de Limoges

Acknowledgements

I would never be doing this PhD without Olivier, it was not always easy mostly because nothing prepare you to start a PhD. Everything is new and you can easily get lost in what you have to do. But Olivier has always been kind and comprehensive no matter how hard were the times, he is a good person[1]. I hope we will be able to work a bit together in the future. I also would like to thank Hieu for the time he took to help me figure out how to express my ideas and the work we have done together. Let's not forget Philippe which played many roles even before the start of this PhD. He puts me in relation with Olivier, helped me figure out what I wanted to do in terms of research and directed my PhD for my first year. He also explained to me why I should not worried about a reject about a paper. I want to thank Céline for the work we have done together and the help she gave me for my different presentations.

I also thanks the reviewers of my thesis Sébastien Canard and Damien Vergnaud for their interesting comments that helped me in improving this document.

Realizing a PhD is hard. Not only because a work of research is hard but also because it is new. Before it, the studies are quite linear, we learn things that people already discovered, that we never truly doubt. Beginning a work of research is about doubting. Doubting about the papers you read but mostly doubting about yourself. This is very important, the more you doubt the more you criticize your work and the more you improve it. At some point you have to stop doubt your ideas and have faith in them and in you. This is also important because having faith in what you do help to convince other researchers that what you do is good and that it can be published and used.

This crucial point between the doubt and the faith is hard to place. It is even harder when this is your first years of research and you don't know how much you are worth. The problem is that I do not trust myself that much. This is why a lot of people were necessary to my success and I will never thanks them enough for it: my Wife, my parents, my sister, my friends Xavier, Zoé, Mickaël, Mathieu, Maël, Adrien, MG, Joëlle, Tam, Florent. Also sport helped me to end this hard last year so I would like to thanks every people I met in the climbing club Climb Up. Please forgive me if I forgot you in my acknowledgement.

[1] This is for me one of the best compliment

Contents

Chapter 1

Introduction

1.1 History of Cryptography

In this section we will see that the willingness of hiding messages is very old. For ages the systems used to hide a message needed a common knowledge between the two parties that are exchanging messages. We will also see that there exists a kind of cryptography which does not need a common knowledge or a common key to encrypt a message. This kind of cryptography is very young compared to the former one. We call the former: symmetric cryptography or secret-key cryptography, and the latter: asymmetric cryptography or public-key cryptography. For this part we will focus only on encryption techniques e.g. the ways to transform a text such that only few people can understand it. Then we will try to describe the nowadays problems to understand our special needs in cryptography.

1.1.1 Antic Cryptography

The Scytale: one of the most ancient example of cryptography is the use of the tool called the Scytale. It was used between the 10th and 7th century BC in Greece. It is a simple stick with a precise diameter. To encrypt a message the sender will wrap a piece of tissue on the stick and write the message on it (see Figure 1.1). To decrypt the receiver will use his own scytale, wrap the tissue on it and simply read the message. The security of this scheme lies in the diameter of the scytale. The diameter of the scytale can be

Figure 1.1: Scytale Technique (Credits: Wikipedia)

though as a key of encryption and decryption. The problem with this technique is that there is not enough diameters possible to align the letters. Thus an eavesdropper knowing the method of encryption can try all the possible alignments of letter, and find the one giving meaning to the text. It allows to decrypt this message but also all the other encryptions since the size of the scytale never changes.

The Caesar's Cipher: A very known example is the one call the Caesar's Cipher. The idea is to shift the letters of the alphabet by a precise number n (that will be the encryption/decryption key). It is described in Figure 1.2. To decrypt a message the receiver will do the same operation but shifting in the reverse order. In a sense this technique has the same problem than the previous one: it is possible to test different number for n, since there are only 26 letters in the alphabet. This means that if the method of encryption is known, the scheme is no longer secure. We can say that the size of universe of possible keys is too small. This is not the only problem, for example a letter will always be encrypted in the same letter.

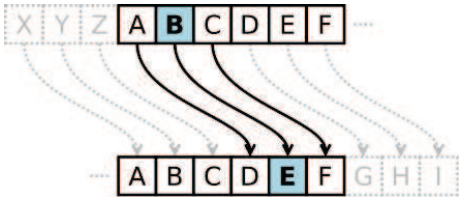


Figure 1.2: Caesar Cipher (Credits: Wikipedia)

1.1.2 Cryptography in the Middle Ages

Kerckhoffs principles: At the end of the 19th century, Auguste Kerckhoffs wrote 6 principles that[1] any cipher system should satisfy. They will guide the future creators to keep in mind what is important to achieve for an encryption scheme.

- the system must be practically, if not mathematically, indecipherable
- it should not require secrecy, and it should not be a problem if it falls into enemy hands
- it must be possible to communicate and remember the key without using written notes, and correspondents must be able to change or modify it at will
- it must be applicable to telegraph communications
- it must be portable, and should not require several persons to handle or operate
- lastly, given the circumstances in which it is to be used, the system must be easy to use and should not be stressful to use or require its users to know and comply with a long list of rules.

[1] in his opinion

The two last principles are less relevant in our world[2] and the third is not respected. The most important is the second one that we could reformulate as: "the security should lie in the knowledge of the key, not in the method of encryption". This principle excludes some kind of schemes like the 2 firsts that we saw because of the small size of the possible "key" universe. Indeed the fact that the size of potential keys set is small allows to guess the key and then the security is not anymore relied on the knowledge of the key. However the next example that does satisfy this second principle[3]:

Vigenere Cipher: This encryption scheme has been created at the end of the 16th century. It is based on the Caesar Cipher with a whole word as key: to encrypt the sender will copy the key word under the text has many times needed to "cover" the whole text. Each letter of the key word represent a number: its position in the alphabet. Each letter of the text will be shifted in the alphabet by the number represented by the letter under it. For example the text *PLAINTEXT* encrypt with the key word *KEY* (e.g. the numbers 11, 5, 25) will be *ZPYSRROBR*. The first letter has been shift by 11, the second one by 5 then 25 then again 11 etc... There is a table helping the encryption in Figure 1.3. The decryption is made by shifting in the other way in the alphabet. This scheme seems as secure as n simple Caesar Cipher, n being the size of the key word. Because the adversary can cut the text in n pieces and then solve n different Caesar cipher. However n is secret so it has to be guessed first[4]. In this encryption scheme it is hard to guess the key because the key can be any word. In the case where the size of the word is equal to the size of the text, the scheme is perfectly secure. It is called the Vernam Cipher.

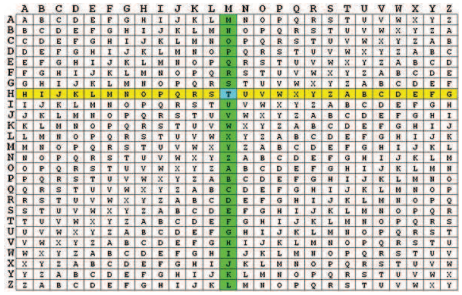


Figure 1.3: Vigenere Cipher (Credits: Wikipedia)

1.1.3 Modern Era

The Enigma Machine: During World War 2 the German army and their allies used a machine called Enigma. This complex machine was easy to use because the sender only had to press the button of the letter he wants to encrypt and the machine responds by enlightening the outputted letter. This machine creates an electrical circuit mapping 2 different letters. The decryption phase is done the same way as the encryption. This machine was very secure because the electrical circuit between the letters change each time a button is pressed.

[2] it still apply in lightweight cryptography for example

[3] Note that the Vigenere Cipher is anterior to the principles of Kerckhoffs, Kerckhoffs might have seen what in this cipher was great and learn from it the second principle

[4] there exist simpler method to break this scheme

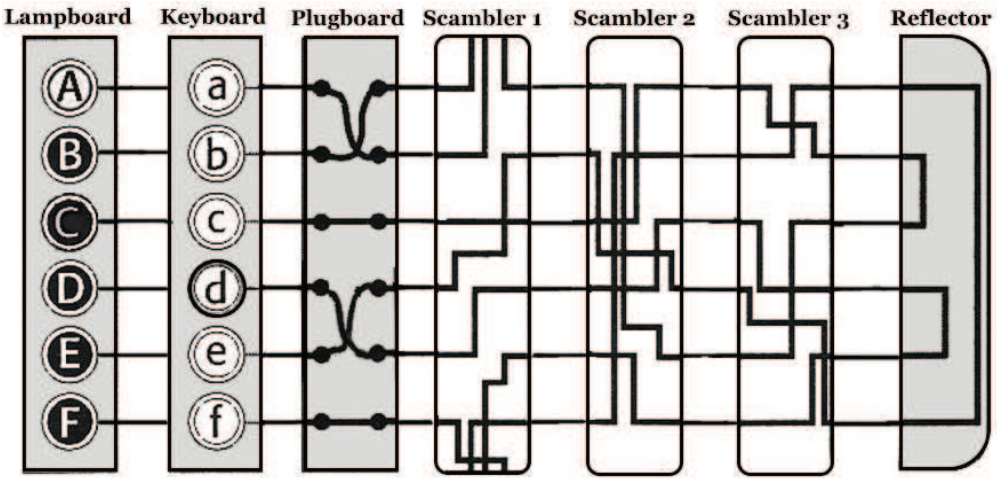


Figure 1.4: The Enigma Machine

The key of encryption is the initial position of the scrambler and choose of the plugboard (see Figure 1.4). Without going into too much details, this machine has been broken due to: first the fact that a letter can not be encrypted in itself decreasing hugely the complexity of any attacks, second the knowledge of a lot of plaintext/ciphertext couple (mostly weather records).

The few examples we have seen are all using the same key for encryption and decryption. This forces the two party to share something before starting exchanging. This is called symmetric cryptography. Finding systems where no secret data are shared before the first exchange was a more challenging problem and find a solution with the Diffie-Hellman cryptosystem in 1976 and the famous RSA system in 1977 [RSA78] [5]. This systems belongs to the public key cryptography.

In this Thesis we look at schemes that go a step further than a simple public key cryptography: they do not need public key to encrypt a message to someone we only need some general public informations and the identity of the receiver. They are called identity-based encryption schemes which are a part of the more general identity-based cryptography.

Today: an era of connections

The last 30 years have seen the rise of Internet. This brings us to consider the digital security as a major problem of this decade and the next ones. This is even more essential with the exponential growth of the Internet of things which forces us to consider a huge number of possible devices constantly connected to Internet. For example, the use of these devices brought DDoS (Distributed Denial of Service) attacks to a whole new level that was not anticipated.

[5] Clifford Cocks created the same kind of system few years before but could not share his work because it was classified by the British Government.

Many aspects of security are desirable. We will list some of them with examples to highlight the central role of the digital security in this new world. We want to ensure the security of the private information of the users. For example we do not want to share our bank card number that we could have enter in an application or an online market account. This example is obvious and so the security of this kind of sensitive data are generally securely handle. There is more concern to have about data that are defined less sensitive. For example the identity of the customer, his name, his address, the name of his close family member or customer related information that could have been harvest like the political opinion, the sexuality ... Those are information that may not seem sensitive. In fact this kind of information can be used in different way that could harm the user through harassment, discrimination ...

We want to ensure the security of the devices/algorithms used on the wild connected world. Indeed an encrypted message should remain secret for every non receiver party. This is this aspect of cryptography that is considered central for the population. Another example which the population does not seem to care a lot about is the possibility of a device to be corrupted. Imagine a device of the Internet of things that could be remotely controlled. This is not a big deal if someone hack a pillow to see how good was the sleep of the owner but it seems more a problem when the device is a car or a medical device. Those are dramatic examples but we can have different interest in taking the control or damaging devices. For example, if the home-thermostat are not secure enough then a gas company could slightly change the temperature measured by the device allowing this company to make bigger profits without anyone noticing.

These examples show how important it is to secure this new world and I hope it somehow responds to the "I have nothing to hide" sentence that we hear too often. So we want to secure this world but as in the real life there is always a trade off between security and freedom. In the context of digital security freedom will be efficiency/speed: the more we secure a simple message we send the more we need time or power of computation to achieve it.

In this Thesis we are going to investigate a very secure aspect of the public key cryptography and thus our solutions will have a little overcost[6] compared to less secure solutions. But we stress that our solutions are practical and can be used with an overcost in term of computations and communications very acceptable (with regard to the level of security achieved). Also we will talk a lot about Identity-based cryptography which will be described in the following. This kind of cryptography solves a lot of problems when there is a lot of users in the system which is the case in our time.

1.2 High Level Definitions

Identity-based Encryption The concept of encryption is understood by most

[6] which can be in terms of computation or communication

of people. An identity-based encryption (IBE) is an encryption scheme where there is no need to know a public key to encrypt for a user. Only the identity of the recipient and some general public information (called a master public key) need to be known by the sender. For example an identity can be a name or a mail address (because of its uniqueness). This particular kind of encryption has many advantages: users do not need to store a public key for each user they might discuss with. If the keys can be hold by a trusted authority in the context of the PKI architecture, in the identity-based context the users do not need to trust this kind of authority any more. The strongest advantage and maybe the most underrated is the fact that IBE schemes allows to compute ciphertexts for a string of bits. We will develop that in this Thesis.

The disadvantages of this kind of encryptions is the difficulty to create them. It took several years between the idea of IBE by Shamir [Sha84] and the realization of this idea [BF01]. It needed a new and heavy tool called pairing to be achieved and at this day there is very few other efficient way to create IBE.

Note that there exists many kind of Identity-based cryptography like Identity-based Signature that we will be used in this Thesis.

Oblivious Transfer (OT) In our research about Identity-based Encryption we will encounter and then study protocols called Oblivious Transfer. An Oblivious Transfer is an interactive way to exchange files in a database with a very high degree of secrecy. In fact a user will be able to get a file from a database without this database knowing which file has been requested. On top of that the database does not want the user to have access to a file he does not query. In the context of the security of protocols we need to be very careful. If the protocol is secure when used as it should be used it can be unsecure when used dishonestly. The UC model [Can01] is the security model that ensure that the security remains the same in both context. Highly secure Oblivious Transfer will be proved secure in the UC model.

1.3 This Work

What was the idea behind this Thesis? Most of the things existing in the world are designed or used for a special purpose. However sometimes people voluntary or involuntary find a new use to this objects. This is not a new thing, for example the flint was used by the first kind of Homo 1.6 millions years ago to cut meat or leather. But approximatively 1.4 millions year after they found out that this same flint could be used to create fire. Once they learned that this new application to the flint was possible, the deep knowledge about the use of a flint helped them to quickly master the creation of fire. This example highlights the fact that using known concepts or tools to create new ones is sometimes a good starting point. Indeed it is easier to work in a familiar environment. Moreover the idea of using anything in a strange way and make it work and sometimes efficient is itself very funny.

This was in fact the true idea behind this Thesis: "How far can we go with" and we focused on the case of "How far can we go with Identity-based Encryption". We got a bit lost in the way when studying Oblivious Transfer itself but it was also an interesting part of the journey. Obviously this Thesis does not set the limits of the use of IBE but knowing where we can go is knowing that the limits are further. Roughly speaking: How far can we go? Well the answer is beyond what we have done here.

What do we achieve in this Thesis?

Downgradable Identity-based Encryption (DIBE) We manage to generalize a whole class of different identity-based encryption schemes. We called this new scheme Downgradable Identity-based Encryption. This is an Identity-based Encryption that enables key delegation with some conditions: a user with the identity id can recover a user secret key of any identity $\tilde{\mathsf{id}}$ where if $\tilde{\mathsf{id}}_i = 1$ then $\mathsf{id} = 1$, we note $\tilde{\mathsf{id}} \preceq \mathsf{id}$. This condition gives some kind of power to bits equal to 1 compared to bits equal to 0. We find out that this special type of hierarchy is present in Attribute-based scheme. Indeed in an Attribute-based Encryption a user is able to decrypt a message if he verifies a policy made of attributes. We can draw a parallel between the role of an attribute and the role of a 1 in an identity, if each attribute is represented by a bit in an identity string then to encrypt for the conjunction of several attribute, we only need to encrypt for the identity made of 0's and having 1's at the position of these attributes. This is our starting point for the Section 4.2. We also detail the transformations from DIBE to Hierarchical IBE, wicked IBE and wildcarded IBE, and an instantiation of DIBE derived from the Hierarchical Identity-based Encryption from [BKP14] in Section 3.1. This work is still in the process of submission.

Identity-based Designated Verifier Signature We use Hierarchical IBE to construct a special type of signature called Identity-based Designated Verifier Signature. In fact we first create a secure and efficient Designated Verifier Signature scheme using implicit decommitment, namely Smooth Projective Hash Functions, to verify a signature without being able to reveal the signer. Then we describe a way to transform it into an Identity-based scheme. This work has been published in the article [BCGJ17] and is detailed in Section 4.1

Oblivious Transfer from Blind IBE First we find a way to securely and efficiently transform any IBE into a blind IBE described in Section 3.2, in this context a user can ask for any secret key without the authority knowing which identity is the target of this request. This transformation is done using Smooth Projective Hash Functions allowing user to commit a request that will be implicitly decommit to answer the request. This technique is even more efficient with affine IBE. Using this new efficient primitive we are able to instantiate an adaptive UC-secure Oblivious Transfer: first the database is encrypted with an IBE, each line is encrypted for the identity equal to the bit string corresponding to the number of the line we are looking at. Then it is sent to the user and

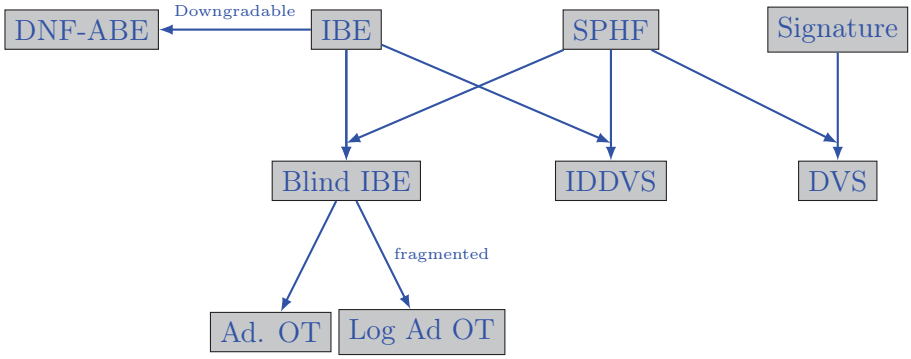


Figure 1.5: Relations Between Primitives in this thesis

then the user asks for user secret key as for the Blind IBE scheme. There are some tricks to achieve the UC-security and decrease communication costs, it is described in section 4.3. This work has been published in the article [BCG16].

Oblivious Transfer Generalization We are able to generalize different schemes such as Oblivious Transfer and Oblivious Signature-based Envelope into the concept of Oblivious Language-based Envelope and even give a general UC-secure instantiation. This new protocol allows a user to recover one or several informations if it owns a word in one or several languages. Depending on the type of language this scheme can instantiate different protocols. Our instantiation also makes use of Smooth Projective Hash Functions. This work has been published in [BCG16].

Very Efficient Oblivious Transfer from Password Authenticated Key Exchange During the study of Oblivious Transfer we find out that it was possible to instantiate it using Password Authenticated Key Exchange. We apply and optimize this transformation to multiple PAKE and the one from [JR15] gave the most efficient UC-secure Oblivious Transfer. This work is described in Section 5.2.3 and has been published in [BCG17].

Outline In the second chapter we will first describe the different primitives/tools that we will need for our work. Each time we will define the notion, define the security requirement and give examples. The next chapter is about the work on the Identity-based Primitives as a stand alone. After that we will use this work and apply it. The last chapter is about Oblivious Transfer: generalizations on one side and efficient instantiation from PAKE on the other side.

We summarize all the link between primitives that we found in this thesis in Figure 1.5.

Chapter 2

Definitions And Preliminaries

2.1 Pairing Based Cryptography

2.1.1 Diffie-Hellman Problems

In provable cryptography, the security relies on supposedly hard assumptions. In other words we say that a scheme is secure if the underlying hard assumption holds. There are many candidates of hard assumptions, a notable one is the hardness of the discrete logarithm (DL) problem: let $\mathbb{G}$ be a group of order q and g, h two elements of this group. The discrete logarithm of h in basis g is the element $a \in \mathbb{Z}_q$ such that $g^a = h$[1]. The discrete logarithm problem is to recover a knowing g and h.

Another notable assumption is the Computational Diffie Hellman assumption (CDH) [DH76]. Given a group $\mathbb{G}$ and 3 elements of this group: (g, g^a, g^b). It is hard to compute g^{ab}. The decisional version of this assumption states that it is hard to distinguish g^{ab} from a random element. An adversary solving the computational problem can trivially solve the decisional one. More formally:

Definition 1 (Decisional Diffie-Hellman Assumptions DDH)**.** *We say that the Decisional Diffie-Hellman Assumption holds relative to $(\mathbb{G}, g)$ a group and its generators. if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{D}$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, g^a, g^b, g^{ab} = 1] - \Pr[\mathcal{D}(\mathcal{G}, g^a, g^b, h) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over the group generation algorithm, $(a, b) \xleftarrow{\$} \mathbb{Z}_p$[2]*, $h \xleftarrow{\$} \mathbb{G}$.*

Solving the discrete logarithm problem allows to solve the Diffie-Hellman (DH) assumptions thus DH assumptions is weaker than the DL one. In other words a scheme secure under the DL assumption is also secure under the DH assumption. In this thesis we will use a lot of variants of the decisional Diffie-Hellman assumption such as the matrix Diffie-Hellman or the bilinear DH (BDH).

[1] if it exists, it is sure if g is a generator of $\mathbb{G}$

[2] $\mathbb{Z}_p$ is the group $\mathbb{Z}/p\mathbb{Z}$ with p a prime number

Before defining these variants we first recall the notion of pairings. Pairings are one of the most powerful tool used in modern cryptography. Roughly speaking, it allows to compute the Diffie Hellman tuple but with fourth element in another group.

Definition 2 (Pairing)**.** *A pairing e is a bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$ (where $\mathbb{G}_i, i \in \{1, 2, T\}$ are groups) and must be:*

- *Bilinear: $e(g_1^a, g_2^b) = g_T^{ab}$ with $g_T = e(g_1, g_2)$ and for all $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}$*
- *Non degenerate: the image of e should not be reduce to $\{1_{\mathbb{G}_T}\}$. If $\mathbb{G}_1$ and $\mathbb{G}_2$ are prime order group and g_1, g_2 are generators then it means that: $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.*
- *Computable: it should exist an polynomial time algorithm able to compute the pairing for any $(g_1, g_2) \in \mathbb{G}_1 \times \mathbb{G}_2$.*

2.1.2 Security Assumption: Pairing Groups and Matrix Diffie-Hellman Assumption.

For simplicity when we will use different groups, we will use implicit representation of group elements as introduced in [EHK+13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (this correspond to solving the discrete logarithm problem in $\mathbb{G}_s$). Moreover, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and e a pairing. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

Definition 3 (Bilinear Diffie-Hellman problem (BDH))**.** *The decisional bilinear Diffie-Hellman problem is to distinguish two distributions:*

$([a], [b]_1, [c]_2, [abc]_T)$ *and* $([a]_1, [b]_1, [c]_2, h)$ *where* $\mathcal{G} \leftarrow \mathsf{GGen}(1^{\mathfrak{K}})$ *and* $h \xleftarrow{\$} \mathbb{G}_T$.

More generally, for a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{Z}_p^{n \times m}$ we define $[\mathbf{A}]_s$ as the implicit representation of $\mathbf{A}$ in $\mathbb{G}_s$:

$$[\mathbf{A}]_s := \begin{pmatrix} g_s^{a_{11}} & \dots & g_s^{a_{1m}} \\ & & \\ g_s^{a_{n1}} & \dots & g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n \times m}$$

For $a, b \in \mathbb{Z}_p^k$, $e([a]_1, [b]_2) := [a^\top b]_T \in \mathbb{G}_T$. We define the matrix Diffie-Hellman (MDDH) assumption [EHK+13].

Definition 4 (Matrix Distribution)**.** *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution an algorithm that outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $A \xleftarrow{\$} \mathcal{D}_k$[3] form an invertible matrix, we denote this matrix $\overline{A}$, while the last line is denoted $\underline{A}$. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([A], [Aw])$ and $([A], [u])$ where $A \xleftarrow{\$} \mathcal{D}_k$, $w \xleftarrow{\$} \mathbb{Z}_p^k$ and $u \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. Formally:

Definition 5 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH)**.** *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to* GGen *in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [\mathbf{A}bw]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\mathbf{A}]_s, [bu]_s) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, bw \xleftarrow{\$} \mathbb{Z}_p^k, bu \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

For each $k \geq 1$, [EHK+13] specifies distributions $\mathcal{L}_k$, $\mathcal{U}_k$, ... such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. For example the distributions are for $k = 2$, where $a_1, \ldots, a_6 \xleftarrow{\$} \mathbb{Z}_p$.

$$\mathcal{L}_2 : \mathbf{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \mathbf{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

It was also shown in [EHK+13] that $\mathcal{U}_k$-MDDH is implied by all other $\mathcal{D}_k$-MDDH assumptions.

Lemma 1 (Random self reducibility [EHK+13])**.** *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-MDDH is random self-reducible. In particular, for any $m \geq 1$ and for all PPT adversaries $\mathcal{D}$ and $\mathcal{D}'$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) + \tfrac{1}{q-1} \geq \mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}')$$

where $\mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}') := \Pr[\mathcal{D}'(\mathcal{G}, [\mathbf{A}], [\mathbf{AW}]) \Rightarrow 1] - \Pr[\mathcal{D}'(\mathcal{G}, [\mathbf{A}], [\mathbf{U}]) \Rightarrow 1]$, with $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\mathbf{A} \xleftarrow{\$} \mathcal{D}_k, \mathbf{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times m}, \mathbf{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times m}$.

Remark: It should be noted that $\mathcal{L}_1, \mathcal{L}_2$ are respectively named semi-external Diffie Hellman[4] (SXDH) and Decision Linear (DLin) [EHK+13] assumptions. $\mathcal{L}_1$ is generated using a scalar as matrix.

[3] the notation $\xleftarrow{\$}$ means that the left element is chosen randomly in the set at the right of the arrow

[4] in other terms this assumption state that DH holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$

2.1.3 Security Models

When rigorously defining security of protocols we have different ways to link the security and the hardness assumption. We will here describe the most common ways.

Random Oracle Model In the random oracle model (ROM), we assume that a public accessible algorithm (often a hash function) can be modelled as a random oracle. It means that any adversary cannot distinguish the output of the algorithm from a distribution uniformly chosen at random element of the output domain. In [CCG+94] they explained that there exists no true random oracle. Moreover when instantiated some signature or encryption scheme which where secure in the ROM, it leads to truly insecure scheme. The ROM can gives a reasonable level of security. However the high level research tend to avoid it.

Standard Model In opposition to random oracle model, the desirable model of security is the standard model. In the standard model the security is only based on the complexity assumptions. It means that the adversary is only limited by the time and computational power available.

Universally Composable (UC) Framework. The goal of the UC framework [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial time adversary $\mathscr{A}$ (which controls the communication between the players), a simulator $\mathscr{S}$ such that no polynomial time environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathscr{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathscr{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathscr{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

Simple UC Framework. Canetti, Cohen and Lindell formalized a simpler variant in [CCL15], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated; There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions.

Advantage A common notion when discussing about security games and adversaries playing these games is the advantage. This is a simple notion that we

describe in the following definition.

Definition 6. *The advantage of an adversary $\mathcal{A}$ in winning the experiment* Exp *is the probability that $\mathcal{A}$ wins the experiment minus* $1/2$.

2.2 Parameters and Efficiency

When there is an efficiency comparison between different schemes/protocols we often compare them in terms of communication cost. This communication cost is expressed in number of group elements.

We have to be careful with this approach, indeed the structure/properties of the groups involved change a group element's size of storage. The first thing to take into account is the cardinal of the group, there is a ratio superior to 12 between the size of an element in a prime order group and an element in a composite order group (according to the National Institute of Standards and Technology [Gui13]).

Computational time is also a very important parameter to describe the efficiency. Once again, the type of group used imply different time performance: for the same 126 bit security, a pairing computation takes 254 more time in a composite order group than in a prime order group and exponentiation is more than 300 time slower. This is quite problematic because pairing computations already take a lot of time. All these information can be found in [EMJ17].

The size of group elements depends on the security we want but also on the security reduction from the hard assumption we have. Thus the more direct is the reduction the less group elements' size increases.

2.3 Basis Of Cryptography: Encryption

We will formally define what is an encryption[5] scheme and what kind of security requirement it must follow.

Definition: an encryption scheme $\mathcal{E}$ is described through four PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key ek and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, M; \rho)$ outputs a ciphertext $\mathcal{C}$, on a plaintext M, under the encryption key ek, using the randomness ρ;
- $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C})$ outputs M, encrypted in the ciphertext $\mathcal{C}$ or $\perp$.

[5] Here we obviously define a public key encryption scheme

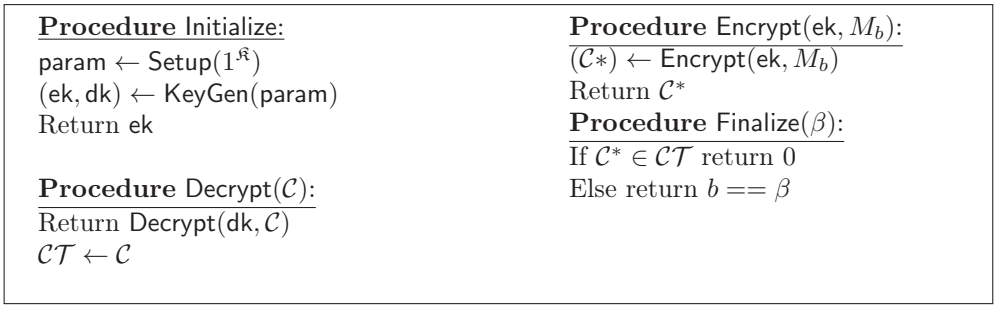


Figure 2.1: IND-CCA-security game

Security: such encryption scheme is required to have the following security properties:

- *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every random ρ, we should have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, M; \rho)) = M$.
- *Indistinguishability under Adaptive Chosen Ciphertext Attack* IND-CCA [NY90, RS92]:
  We say that the encryption scheme is IND-CCA secure if the advantage[6] of an adversary in winning IND-CCA-security game Figure 2.1 is negligible.
  Description of the game in Figure 2.1: a challencger $\mathcal{C}$ will create a game for the adversary $\mathcal{A}$.
  - **Procedure** Initialize: $\mathcal{C}$ will honestly compute the parameters and the pair of keys and return ek to $\mathcal{A}$.
  - **Procedure** Decrypt($\mathcal{C}$): $\mathcal{A}$ can request at any time decryption of ciphertexts, the ciphertext decrypted are stored (if the adversary has not access to the procedure Decrypt the security is called IND-CPA (Indistinguishability under Chosen Plaintext Ciphertext Attack)).
  - **Procedure** Encrypt(ek, M_b): $\mathcal{A}$ will submit two messages M_0 and M_1, in game b ($\in \{0, 1\}$) the challenger encrypt the message M_b and return the ciphertext to $\mathcal{A}$.
  - **Procedure** Finalize(β): $\mathcal{A}$ submit a bit β and win if it never decrypt the ciphertext $\mathcal{C}^*$ and if M_β is encrypted in $\mathcal{C}^*$.

Examples: One of the most simple example is the ElGamal encryption [ElG84] which is IND-CPA secure under DDH assumption. It is described in Figure 2.2.

The linear encryption described in Figure 2.3 has been introduced by Boneh Boyen and Shacham in [BBS04] is IND-CPA-secure under DLin.

In the following of the thesis we will need another encryption called the Cramer-Shoup encryption scheme [CS98] which is an IND-CCA version of the ElGamal Encryption. We present it here as a labeled public-key encryption scheme in

[6] see definition 6

- $\mathsf{Setup}(1^{\mathfrak{K}})$: Generates a group $\mathbb{G}$ of prime order p, with a generator g, $\mathsf{param} = (\mathbb{G}, p, g)$.
- $\mathsf{KeyGen}(\mathsf{param})$: Picks $h \xleftarrow{\$} \mathbb{G}$,$\mathsf{dk} = x \xleftarrow{\$} \mathbb{Z}_p$, and sets $\mathsf{ek} = g^x$.
- $\mathsf{Encrypt}(\mathsf{ek}, M \in \mathbb{G}; r)$: Outputs $\mathcal{C} = (M \cdot \mathsf{ek}^r, g^r)$
- $\mathsf{Decrypt}(\mathsf{dk}, C)$: Outputs $\mathcal{C}_1/(\mathcal{C}_2^{\mathsf{dk}})$

Figure 2.2: ElGamal Encryption

- $\mathsf{Setup}(1^{\mathfrak{K}})$: generates a group $\mathbb{G}$ of prime order p, with a generator g, $\mathsf{param} = (\mathbb{G}, p, g)$.
- $\mathsf{KeyGen}(\mathsf{param})$: $\mathsf{dk} = (y_1, y_3) \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathsf{ek} = (\mathsf{ek}_1, \mathsf{ek}_2, \mathsf{ek}_3) := (g^{y_1}, g, g^{y_3})$
- $\mathsf{Encrypt}(\mathsf{ek}, M)$: $(r_1, r_2) \xleftarrow{\$} \mathbb{Z}_p^2$, $\mathcal{C} := (\mathsf{ek}_1^{r_1}, \mathsf{ek}_2^{r_2}, \mathsf{ek}_3^{r_1+r_2} \cdot M)$
- $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C})$: outputs $\mathcal{C}_3/(\mathcal{C}_1^{y_3/y_1}\mathcal{C}_2^{y_3})$

Figure 2.3: Linear Encryption

Figure 2.4, the classical version is done with the label space set to $\emptyset$[7].The security of the scheme is proven under the DDH assumption and the hash function used is a Universal One-Way Hash Function. A generalization of this encryption to the $k - \mathsf{MDDH}$ assumption can be found in figure 2.5.

[7]A labeled public-key encryption scheme is a set of public-key encryption schemes. The choose of the label make the encryption scheme unique. Two labels gives two different encryption schemes

- $\mathsf{Setup}(1^{\mathfrak{K}})$: generates a group $\mathbb{G}$ of order p, $\mathsf{param} = (\mathbb{G}, p)$.
- $\mathsf{KeyGen}(\mathsf{param})$: generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$ Chooses a Universal One-Way hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (see def 14) $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M \in \mathbb{G}; r)$: Outputs $\mathcal{C} = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^{\xi})^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C})$: Computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If True computes $M = e/(u_1^z)$ and outputs M. Otherwise, outputs $\perp$.

Figure 2.4: Cramer Shoup Encryption

- $\mathsf{Setup}(1^{\mathfrak{K}})$: generates groups $\mathbb{G}_1$ of order p with generator g_1, $\mathsf{param} = (\mathbb{G}, p)$.
- $\mathsf{KeyGen}(\mathfrak{K})$: Picks $E \xleftarrow{\$} \mathcal{D}_k, u, v \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, sets $\mathsf{ek} = ([E]_1, [u^\top E]_1, [v^\top E]_1), \mathsf{dk} = (\underline{E} \cdot \overline{E^{-1}}, u, v)$ to be the public encryption key, where $\mathfrak{H}_K$ is a random collision-resistant hash function from $\mathcal{H}$ [a].
- $\mathsf{Encrypt}(\mathsf{ek}, M \in \mathbb{G}_1, \ell; \boldsymbol{\mu})$ If $M = g_1^m$, $\mathcal{C} = (e = \mathsf{ek}_1\boldsymbol{\mu} + \begin{bmatrix} 0 \\ m \end{bmatrix}_1, w = [(\mathsf{ek}_2 + \theta \mathsf{pk}_3)\boldsymbol{\mu}]_1)$. where $\theta = \mathfrak{H}_K(\ell, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C})$ If $[w]_1 \stackrel{?}{=} [(u^\top + \theta v^\top)\overline{e}]_1$, then outputs $M = \mathsf{sk}\overline{e} + \underline{e}$. Otherwise output $\perp$.

[a] Like Cramer-Shoup one could rely on an universal one-way hash function family instead see def 14

Figure 2.5: Cramer Shoup Encryption under $k - \mathsf{MDDH}$

2.4 Digital Signatures

2.4.1 Definition and Security

A digital signature scheme $\mathcal{S}$ [DH76, GMR88] allows a signer to produce a verifiable proof that proves the authenticity of a message. It is described through four algorithms:

Definition 7 (Digital Signature Scheme)**.** $\sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param *of the scheme, for example the message space;*
- $\mathsf{KeyGen}(\mathsf{param})$*, outputs a pair of* $(\mathsf{sk}, \mathsf{vk})$*, where* sk *is the (secret) signing key, and* vk *is the (public) verification key;*
- $\mathsf{Sign}(\mathsf{sk}, M; \mu)$*, outputs a signature* σ*, on a message* M*, under the signing key* sk*, and some randomness* μ*;*
- $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ *checks the validity of the signature* σ *with respect to the message* M *and the verification key* vk*. And so outputs a bit.*

In the following we will expect at least two properties for signatures:

- *Correctness*: For every pair $(\mathsf{vk}, \mathsf{sk})$ generated by KeyGen, for every message M, and for all randomness μ, we have $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M; \mu)) = 1$.

- *Existential Unforgeability under Chosen Message Attacks [GMR88]* ($\mathsf{EUF}-\mathsf{CMA}$). We say that the signature scheme is $\mathsf{EUF}-\mathsf{CMA}$ secure if the advantage of an adversary in winning $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}$ is negligible. The oracle OSign sign a message and store any message queried in $\mathcal{SM}$.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\mathsf{sk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5. If $m^* \in \mathcal{SM}$ or $b \neq 1$ output 0
6. Else output 1

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = 1], \quad \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(\mathfrak{K}, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}).$$

There is an interesting primitive close to signature but in a symmetrical cryptography setup, this is the message authentication code (MAC)[8].

Definition 8 (Message Authentication Code)**.** *A* MAC *is described through three algorithms* $(\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Ver})$*:*

- $\mathsf{Gen}_{\mathsf{MAC}}(\mathsf{param})$*: Returns* $\mathsf{sk}_{\mathsf{MAC}}$*.*
- $\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, m)$*: Returns a tag* τ *on a message* m *from the message space* $\mathcal{M}$*.*
- $\mathsf{Ver}(\mathsf{sk}_{\mathsf{MAC}}, m, \tau)$*: Returns a verification bit* b*.*

The MAC *is correct if for any correctly computed tag* τ *the verification bit outputted by* Ver *is equal to* 1*. For the security we define is unforgeability under Chosen Message Attacks as for signature schemes.*

2.5 Identity-based Encryption

The most studied Identity-based primitive is by far the Identity-based Encryption (IBE). In this part we will describe formally what is an IBE and what kind of security is required for this type of scheme. Further we will see how the building of such schemes has changed through time, then we will see a very recent example which will be a basis of our work. Finally we will talk about some features that will be very important to use IBE in unexpected ways.

2.5.1 Definition and Security

Identity-based encryption was first introduced by Shamir in [Sha84] who was expecting an encryption scheme where no public key will be needed for sending a message to a precise user, defined by his identity. Thus any user wanting to send a private message to a user only needs this user's identity and a master public key. It took 17 years for the cryptographic community to find a way to

[8] We will use them in the proof of security of the DIBKEM in section 3.1

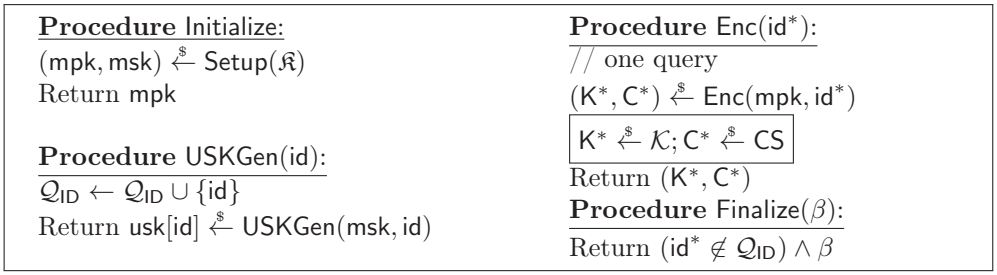


Figure 2.6: PR-ID-CPA-security: Security Games $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{real}}$ and $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}$ (boxed).

realize this idea. The first instantiation was proposed in [BF01] by Boneh and Franklin.

In this thesis we will consider the slightly different concept of identity-based key encapsulation mechanism (IBKEM). In an IBKEM the sender does not choose a message to encrypt. The encryption algorithm will output a key and a ciphertext. The recipient will recover the key by decrypting the ciphertext.

Formally an IBKEM scheme IBKEM consists of four algorithms $\mathsf{IBKEM} = (\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$. Every IBKEM can be transformed into an ID-based encryption scheme IBE using a (one-time secure) symmetric cipher.

Definition 9 (Identity-based Key Encapsulation Scheme)**.** *An identity-based key encapsulation scheme* IBKEM *consists of four PPT algorithms* $\mathsf{IBKEM} = (\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*: from the security parameter* $\mathfrak{K}$*, returns the (master) public/secret key* $(\mathsf{mpk}, \mathsf{msk})$*. We assume that* mpk *implicitly defines an identity space* ID*, a key space* $\mathcal{K}$*, and ciphertext space* CS*.*
- $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$*: returns the user secret-key* $\mathsf{usk}[\mathsf{id}]$ *for identity* $\mathsf{id} \in \mathsf{ID}$*.*
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*: returns a symmetric key* $\mathsf{K} \in \mathcal{K}$ *together with a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to identity* id*.*
- $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$*: returns the decapsulated key* $\mathsf{K} \in \mathcal{K}$ *or the reject symbol* $\perp$*.*

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{mpk}, \mathsf{msk})$ *generated by* $\mathsf{Setup}(\mathfrak{K})$*, all identities* $\mathsf{id} \in \mathsf{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$ *and all* (K, C) *output by* $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*:* $\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{K}] = 1$.

The security requirements for an IBKEM we consider here are indistinguishability and anonymity against chosen plaintext and identity attacks we define pseudorandom ciphertexts against chosen plaintext and identity attacks (PR-ID-CPA) which means that challenge key and ciphertext are both pseudorandom. We define PR-ID-CPA-security of IBKEM formally via the games given in Figure 2.6.

Definition 10 (PR-ID-CPA Security)**.** *An ID-based key encapsulation scheme* IBKEM *is* PR-ID-CPA*-secure if for all PPT* $\mathscr{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{IBKEM}}(\mathscr{A}) := |\Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{rand}} \Rightarrow 1]|.$$

There are many different definitions of security for IBE each one corresponding to different power given to the adversary. Of course the more power we give to the adversary in the security definition the more secure is our scheme.

An interesting definition of security, is the selective-ID security definition. The only difference with the one from Figure 2.6 is that the challenge identity has to be output by the adversary before the procedure $\mathsf{USKGen}(\mathsf{id})$. This is a weaker security definition. It may looks like a detail but in fact there are a lot of problems when moving from selective-ID security to full security. For example in [BB04] the authors state that a selectively secure IBE is fully secure as soon as we can have a collision resistant hash function[9] from the identity space to bits string of size $d \in \mathbb{Z}$. But the reduction is not tight and introduces a factor lose 2^d. As well as in the case of the scheme [BBG05] using the techniques of Waters [Wat05], the authors have to add a collision resistant hash function and they loss an exponential factor in the security parameters (here 2^{nd} where n is the hierarchy depth).

Remark 1. *For simplicity when talking about IBKEM and IBE we will often only refer to IBE. This is not such a problem since it is easy to have one from the other:*

- *IBE to IBKEM: choose* $\mathsf{K} \xleftarrow{\$} \mathcal{K}$ *and set the output of* $\mathsf{Enc}_{\mathsf{IBKEM}}(\mathsf{mpk}, \mathsf{id})$ *to* $(\mathsf{C}, \mathsf{K}) = (\mathsf{Enc}_{\mathsf{IBE}}(\mathsf{mpk}, \mathsf{id}, \mathsf{K}), \mathsf{K})$.
- *IBKEM to IBE: set the output of* $\mathsf{Enc}_{\mathsf{IBE}}(\mathsf{mpk}, \mathsf{id}, M)$ *to* $(M \oplus \mathsf{K}, \mathsf{C})$ *where* $(\mathsf{C}, \mathsf{K}) = \mathsf{Enc}_{\mathsf{IBKEM}}(\mathsf{mpk}, \mathsf{id})$*. Also set the output of* $\mathsf{Dec}_{\mathsf{IBE}}$ *to* $(M \oplus \mathsf{K}) \oplus \mathsf{Dec}_{\mathsf{IBKEM}}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$.

The other algorithms stay the same.

2.5.2 IBE's State of The Art

In this section we will describe some IBE schemes and see what has changed during these years of research on IBE. We will describe two historical schemes and talk about the main other ones without describing them completely. As said before the first IBE scheme was the one from Boneh and Franklin in [BF01]. Their scheme is described in Figure 2.7

This IBE is secure in the random oracle model, both hash functions are modelled as random oracles. As you can see this work is based on pairings. It has to be

[9] see definition 14

$\underline{\mathsf{Setup}(\mathfrak{K})}$
$(p, \mathbb{G}, \mathbb{G}_T, e, g) \xleftarrow{\$} \mathsf{GGen}_s(1\mathfrak{K})$
Choose H (resp. H_T) two hash functions from $\{0,1\}^*$ to $\mathbb{G}$ (resp.$\mathcal{M}$)
$s \xleftarrow{\$} \mathbb{Z}_p^*, h \leftarrow g^s$
Return
$(\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, H, H_T, e, h),$
$\mathsf{msk} = s)$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$
Return $\mathsf{usk}[\mathsf{id}] = H(\mathsf{id})^s$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, M \in \mathcal{M})}$
Return $\mathsf{C} = (g^r, M \oplus H_T(e(H(\mathsf{id}), h)^r))$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$
Set C as $(\mathsf{C}_1, \mathsf{C}_2)$
Return $M = \mathsf{C}_2 \oplus H_T(e(\mathsf{usk}[\mathsf{id}], \mathsf{C}_1))$

Figure 2.7: Boneh Franklin IBE

noticed that in the same time Cocks[10] in [Coc01] created an IBE based on quadratic residue.

The first IBE scheme secure in the standard model was the one from Boneh and Boyen [BB04]. It is secure under the bilinear Diffie-Hellman problem. We describe it in Figure 2.8.

$\underline{\mathsf{Setup}(\mathfrak{K})}$
$(p, \mathbb{G}, \mathbb{G}_T, e, g_1) \xleftarrow{\$} \mathsf{GGen}_s(1^{\mathfrak{K}})$
$g_2 \xleftarrow{\$} \mathbb{G}$, $U = \{u_{i,j}\}_{i \in \{1,n\}, j \in \{0,1\}} \xleftarrow{\$} \mathbb{G}^{2n}$
Choose $s \xleftarrow{\$} \mathbb{Z}_p^*, h \leftarrow g_1^s$ and k a hash function key in the appropriate key space
Return
$(\mathsf{mpk} = (\mathbb{G}, \mathbb{G}_T, e, U, k, g_1, h, g_2), \mathsf{msk} = g_2^s)$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$
Compute $a = (a_1, ..., a_n) = \mathfrak{H}_K(\mathsf{id}) \in \{0,1\}^n$
Choose $(t_1, ..., t_n) \in \mathbb{Z}_p^n$
Return
$\mathsf{usk}[\mathsf{id}] = (g_2^s \cdot \prod_{i=1}^n u_{i,a_i}^{t_i}, g_1^{t_1}, ..., g_1^{t_n})$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, M \in \mathcal{M})}$
Compute $a = (a_1, ..., a_n) = \mathfrak{H}_K(\mathsf{id}) \in \{0,1\}^n$
Choose $r \xleftarrow{\$} \mathbb{Z}_p$
Return
$\mathsf{C} = (e(h, g_2)^r \cdot M, g^r, u_{1,a_1}^r, ..., u_{n,a_n}^r)$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$
Set C as $(A, B, C_1, ..., C_n)$
And $\mathsf{usk}[\mathsf{id}]$ as $(\mathsf{usk}_0, ..., \mathsf{usk}_n)$
Return $M = A \cdot \frac{\prod_{j=1}^n e(C_j, \mathsf{usk}_j)}{e(B, \mathsf{usk}_0)}$

Figure 2.8: Boneh Boyen IBE

This IBE is secure in the standard model and under a classical assumption (BDH). However this scheme is unpractical because of the size of the ciphertext: it is logarithmic in the number of identities. The first practical scheme by

[10] Once again Clifford Cocks could not share his work because it was classified by the British government

Waters in [Wat05] brings a new construction which is efficient and fully secure in the standard model under a classical assumption. The only drawback was the size of the public key which was linear in the security parameter. In [Wat09] Waters fixes this problem with new techniques in the proof called Dual system encryption. More than bringing a new scheme of IBE the dual system encryption will allow a lot of new works in IBE and Attribute-based Encryption.

2.5.3 Instantiation From [BKP14]

Through this thesis we needed an efficient IBE scheme with particular properties (in particular without hash functions). The natural choice was to use the one from [BKP14]. It has many advantages such as the tightness security[11] to a classical assumption and the fact that the group order is prime. This scheme is following the steps of the one from [Wat09] and use Dual system encryption techniques to prove its security. Note that it is actually an IBKEM scheme.

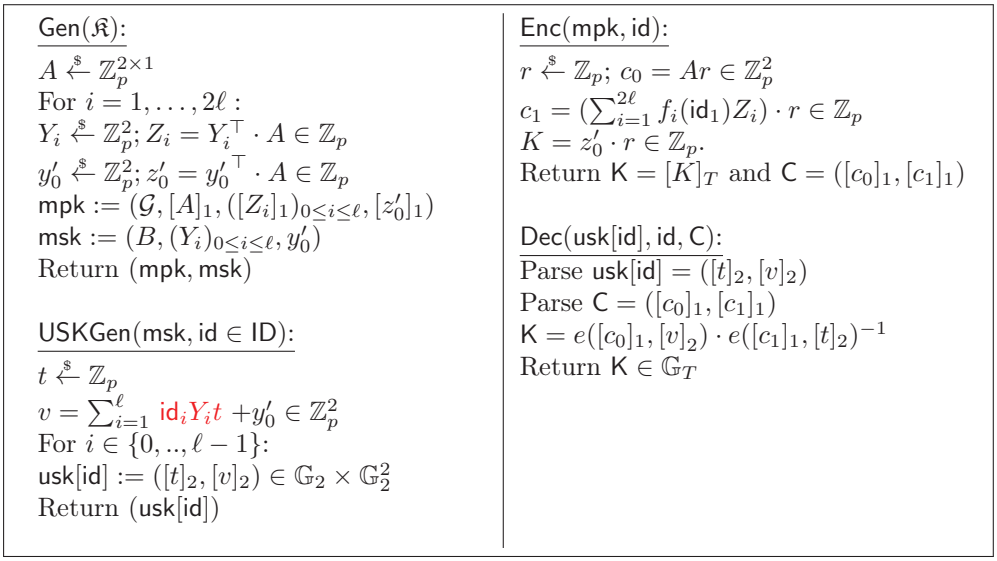


Figure 2.9: Tight IBKEM under SXDH from [BKP14]

Today there is still no practical IBE without pairing. There are some attempt, for example an IBE using garbled circuit [DG17], learning with error problem [GPV08, CHKP10, ABB10] or code-based cryptography [GHPT17] but nothing as practical as pairings. Still it could be very interesting to construct efficient IBE without pairing because pairing computations take time and need an important amount of power of computing. For example in light weight cryptography the use of pairing computations is avoided.

[11] This means that the reduction lost is minimal, to achieve the same level of security we will be able to have shorter parameters

- Wat.$\mathsf{KeyGen}(\mathsf{param})$: $(\mathsf{pk}_1, \mathsf{sk}_1) = (g^z, h^z)$, $z \xleftarrow{\$} \mathbb{Z}_p$
- Wat.$\mathsf{Sign}(\mathsf{sk}_1, \mathcal{M})$: chooses $s \in \mathbb{Z}_p$ and computes $\sigma := (\mathsf{sk}_1 \mathcal{F}(\mathcal{M})^s, g^s)$. Please note that σ is composed of two parts (σ_1, σ_2).
- Wat.$\mathsf{Verify}(\mathsf{pk}_1, \mathcal{M}, \sigma)$: checks $e(g, \sigma_1) = e(\sigma_2, \mathcal{F}(\mathcal{M})) \cdot e(\mathsf{pk}_1, h)$, $\mathcal{F}(\mathcal{M}) := u_0 \prod_{i \in [\![1,k]\!]} u_i^{\mathcal{M}_i}$, with $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_k)$ and $\mathcal{M}_i \in \{0, 1\}$

Figure 2.10: Waters Signature

2.5.4 Signature from Identity-based Encryption

As described in [BF01] it is possible to produce a signature scheme from an IBE one. They proved that the security of the signature relies on the security of the IBE scheme involved. The idea is quite simple: the signer will play the role of the authority of the IBE. Thus the public key of the signature scheme will be all public parameters of the IBE. To sign a message M the signer will produce a private key for the identity: M. The verification is easily done by testing whether or not this element is the private key of M (by encrypting a dummy message and decrypting it).

An interesting example of this kind of signature is the one from Waters in [Wat05] coming from the IBE of the same paper. It is secure under the decisional bilinear Diffie-Hellman problem. For the Waters function $\mathcal{F}$, we assume the existence of independent group generators u_i that define the function $\mathcal{F}(\mathcal{M}) := u_0 \prod_{i \in [\![1,k]\!]} u_i^{\mathcal{M}_i}$, with $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_k)$ and $\mathcal{M}_i \in \{0, 1\}$.

This signature verification makes use of pairing computations. Thus this scheme is not very efficient, but since the verification is made with pairing computations it can be used along other schemes using pairings.This will allow implicit verifications (see Oblivious Signature-based Envelope Section 5.1 or Figure 2.13).

2.5.5 Interesting Features and Properties

Hierarchical Identity-based Encryption: Identities can have a special form, this allow to create some kind of hierarchy between them. That is enough to justify the construction of hierarchical IBE (HIBE) which were introduced in [GS02]. We often represent HIBE with a binary tree, each user represents a node in this tree. In these primitives a user possesses a secret key and is able to construct/delegate a new key for every node/identities under him. Once again HIBE can be used as an actual system of encryption with hierarchy or sometimes it can be used in different ways (see Section 3). We recall a formal definition of HIBE in the term of HIBKEM.

Definition 11. *An HIBKEM* HIBKEM *is described through five algorithm (*Gen, USKGen, USKDel, Enc, D

- $\mathsf{Gen}(\mathfrak{K})$ *returns the (master) public/secret key and delegation key* $(\mathsf{pk}, \mathsf{sk}, \mathsf{dk})$.

- $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id})$ *returns a secret key* $\mathsf{usk}[\mathsf{id}]$ *and a delegation value* $\mathsf{udk}[\mathsf{id}]$.
- $\mathsf{USKDel}(\mathsf{dk}, \mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}], \mathsf{id}, \mathsf{id}')$ *returns a user secret key* $\mathsf{usk}[\mathsf{id}|\mathsf{id}']$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{id})$ *returns a symmetric key* K *together with a ciphertext* C.
- $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$ *returns the decapsulated key* K *or the reject symbol* $\perp$.

In our HIBKEM definition we make the delegation key dk and the user delegation key udk[id] explicit to make our constructions more readable.It will help us for redefining the PR-CMA security we are expecting from an HIBKEM. Indeed to prevent trivial win from the adversary in this game we will not allow a win from the adversary if he requested a user secret key which is a prefix of the challenge identity.

The HIBE from [BKP14] will be our favourite instantiation to use HIBE as a tool. Indeed it is a practical scheme with tight security reduction under a classical assumption namely k-MDDH. We describe this scheme in Figure 2.11. Of course these scheme comes from the IBE from the same paper.

Blind IBE A blind IBE is an IBE where the user secret key can be queried without the authority knowing which identity is related to the key. This might seems awkward to construct such primitive and hard to find applications to this, but it has been created in a precise purpose that we will explain in section 4.3.

We continue to follow the KEM formalism by adapting the definition of a Blind IBE scheme given in [GH07] to this setting.

Definition 12 (Blind Identity-Based Key Encapsulation Scheme)**.** *A Blind Identity-Based Key Encapsulation scheme* BlindIBKEM *consists of four PPT algorithms*

$(\mathsf{Setup}, \mathsf{BlindUSKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

- Setup, Enc *and* Dec *are defined as for a traditional* IBKEM *scheme.*
- $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ *is an interactive protocol, in which an honest user* U *with identity* $\mathsf{id} \in \mathsf{ID}$ *obtains the corresponding user secret key* $\mathsf{usk}[\mathsf{id}]$ *from the master authority* $\mathcal{S}$ *or outputs an error message, while* $\mathcal{S}$*'s output is nothing or an error message (*ℓ *is a label and* $\boldsymbol{\rho}$ *the randomness).*

Defining the security of a BlindIBKEM requires two additional properties, stated as follows (see [GH07, pages 6 and 7] for the formal security games):

1. **Leak-free Secret Key Generation** (called Leak-free Extract for Blind IBE security in the original paper): A potentially malicious user cannot learn anything by executing the BlindUSKGen protocol with an honest authority which he could not have learned by executing the USKGen protocol with an honest authority. Moreover, as in USKGen, the user must know the identity for which he is extracting a key.

$\underline{\mathsf{Setup}(1^\kappa)}$ Return $\mathsf{param} \leftarrow \mathsf{GGen}(1^\kappa)$

$\underline{\mathsf{Setup}(\mathsf{param})}$:

$A \xleftarrow{\$} \mathcal{D}_k, B = \bar{A}$

For $i = 0, \ldots, \ell$:

$z_i \xleftarrow{\$} \mathbb{Z}_p^{k+1 \times n}; Z_i = z_i^\top \cdot A \in \mathbb{Z}_p^{n \times k}$

$z' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; Z' = {z'}^\top \cdot A \in \mathbb{Z}_p^{1 \times k}$

$\mathsf{pk} := (\mathcal{G}, [A]_1, ([Z_i]_1)_{0 \le i \le \ell}, [Z']_1)$

$\mathsf{sk} := ((z_i)_{0 \le i \le \ell}, z')$

Return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{USKGen}(\mathsf{sk}, \mathsf{id} \in \mathsf{ID})^a}$:

$t \xleftarrow{\$} \mathbb{Z}_q^n$;

$v = \sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id}) z_i t + z' \in \mathbb{Z}_q^{k+1}$

$S \xleftarrow{\$} \mathbb{Z}_p^{n' \times \mu}; T = B \cdot S \in \mathbb{Z}_p^{n \times \mu}$

$V = \sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id}) Z_i T \in \mathbb{Z}_p^{(k+1) \times \mu}$

For $i = 0, \ldots, \ell$:

$e_i = Z_i t \in \mathbb{Z}_p^{k+1}$; $E_i = Z_i T \in \mathbb{Z}_p^{k+1 \times \mu}$

$\mathsf{usk}[\mathsf{id}] := ([t]_2, [v]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2^{k+1}$

$\mathsf{udk}[\mathsf{id}] := ([T]_2, [V]_2, ([e_i]_2, [E_i]_2)_{i, \mathsf{id}[i]=1})$

$\in \mathbb{G}_2^{n \times \mu} \times \mathbb{G}_2^{(k+1) \times \mu} \times (\mathbb{G}_2^{k+1} \times \mathbb{G}_2^{(k+1) \times \mu})^{\mathsf{Ham}(\mathsf{id})}$

Return $(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}])$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mathsf{id})}$:

$r \xleftarrow{\$} \mathbb{Z}_p^k$

$c_0 = Ar \in \mathbb{Z}_p^{k+1}$

$c_1 = (\sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id}) Z_i) \cdot r \in \mathbb{Z}_p^n$

$K = z'_0 \cdot r \in \mathbb{Z}_p$.

Return $\mathsf{sk} = [K]_T$ and $\mathsf{C} = ([c_0]_1, [c_1]_1)$

$\underline{\mathsf{USKDel}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}})}$:

Note ℓ the size of id, and $\tilde{\ell}$ the size of $\tilde{\mathsf{id}}$

// Delegating the key:

$\hat{v} = v + \sum_{i=\ell+1}^{\tilde{\ell}} f_i(\mathsf{id}') e_i \in \mathbb{Z}_p^k$

$\hat{V} = V + \sum_{i=\ell+1}^{\tilde{\ell}} f_i(\mathsf{id}') E_i \in \mathbb{Z}_p^{k \times \mu}$

// Rerandomization of $(\hat{v}, \hat{V})$:

$s' \xleftarrow{\$} \mathbb{Z}_p^\mu$; $S' \xleftarrow{\$} \mathbb{Z}_p^{\mu \times \mu}$

$t' = t + Ts' \in \mathbb{Z}_p^n$;

$T' = \hat{T} \cdot S' \in \mathbb{Z}_p^{n \times \mu}$

$v' = \hat{v} + \hat{V} \cdot s' \in \mathbb{Z}_p^k$;

$V' = \hat{V} \cdot S' \in \mathbb{Z}_p^{k \times \mu}$

// Rerandomization of e_i:

For $i, \tilde{\mathsf{id}}[i] = 1$:

$e'_i = e_i + E_i s' \in \mathbb{Z}_p^{k+1}$;

$E'_i = E_i \cdot S' \in \mathbb{Z}_p^{(k+1) \times \mu}$

$\mathsf{usk}[\mathsf{id}'] := ([t']_2, [v']_2)$

$\mathsf{udk}[\mathsf{id}'] := ([T']_2, [V']_2, ([e'_i]_2, [E'_i]_2)_{i, \tilde{\mathsf{id}}=1})$

Return $(\mathsf{usk}[\mathsf{id}'], \mathsf{udk}[\mathsf{id}'])$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$:

Parse $\mathsf{usk}[\mathsf{id}] = ([t]_2, [v]_2)$

Parse $\mathsf{C} = ([c_0]_1, [c_1]_1)$

$\mathsf{sk} = e([c_0]_1, [v]_2) \cdot e([c_1]_1, [t]_2)^{-1}$

Return $\mathsf{sk} \in \mathbb{G}_T$

[a] μ is the difference between the maximal size for and identity and the size of the identity id

Figure 2.11: HIBE based on $k - \mathsf{MDDH}$

2. **Selective-failure Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol; Moreover, the authority cannot cause the BlindUSKGen protocol to fail in a manner dependent on the user's choice.

For our applications, we only need a weakened property for blindness:

3. **Weak Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol.

2.6 Non Interactive Zero Knowledge (NIZK) Argument

A Zero Knowledge Argument introduced in [GMR89], is a set of algorithms involving two parties: a prover and a verifier. These algorithms allow the prover to prove the veracity of a statement without leaking any information. We want three properties on this set of algorithms:

- The verifier should be convinced of the statement if it is actually true
- The verifier should not be able to prove a false statement
- A listener should not learn anything from communication listening.

We call these three properties respectively **Completeness**, **Soundness** and **Zero Knowledge**. Let's formally define a NIZK Argument and describe the properties. For simplicity we describe a statement as a word in a (set of) language(s). For example to prove that a user possesses a solution from an equation, the language will be the set of all solutions of this precise equation.

Definition 13 (Non-Interactive Zero Knowledge Argument)**.** *Assuming a randomness distribution* $\mathcal{D}_{\mathsf{param}}$ *(for a public set of parameters* param*) and a set of languages* $\{\mathfrak{L}_\rho\}_\rho$ *parameterized by a randomness* $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ *and associated with a relation* $\mathcal{R}_\rho$ *(meaning that a word* x *belongs to* $\mathfrak{L}_\rho$ *if and only if there exists a witness* w *such that* $\mathcal{R}_\rho(x, w)$ *holds), a NIZK argument* Π *for this set of languages* $\{\mathfrak{L}_\rho\}_\rho$ *consists of five probabilistic polynomial time (PPT) algorithms* $\Pi = (\mathsf{Gen}_{\mathsf{param}}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Sim}, \mathsf{Ver})$ *defined as follows:*

- $\mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$ *returns the public parameters* param*.*
- $\mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$ *returns a common reference string* crs *and a trapdoor* tk*. We assume that* crs *contains the parameters* param*, a description of the language* $\mathfrak{L}_\rho$*.*
- Prove *takes as input* crs*, a word* x *of the language* $\mathfrak{L}_\rho$ *and a witness* w *corresponding to this word. If* $(x, w) \notin \mathcal{R}_\rho$*, it outputs failure. Otherwise, it outputs a proof* π*.*
- Ver *takes as input* crs*, a word* x *and a proof* π*. It outputs a bit* b *(either* 1 *if the proof is correct, or* 0 *otherwise).*
- Sim *takes as input* crs*, a trapdoor* tk *and a word* x*. It outputs a proof* π *for* x *(not necessarily in* $\mathfrak{L}_\rho$*).*

These algorithms must satisfy the following properties

Perfect Completeness: *for all adversary* $\mathcal{A}$*, all security parameter* $\mathfrak{K}$*, all public parameters* $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$*, all randomness* $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ *and all* $(x, w) \in \mathcal{R}_\rho$*, the following holds:*

$$\Pr[(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathsf{Ver}(\mathsf{crs}, x, \pi) = 1] = 1$$

Perfect Soundness: *For a given security parameter $\mathfrak{K}$, a scheme achieves soundness if for all adversary $\mathcal{A}$, we have*

$$\Pr[\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K}); \rho \leftarrow \mathcal{D}_{\mathsf{param}}; (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); (x, \pi) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{crs}, \rho) : x \notin \mathfrak{L}_\rho \wedge \mathsf{Ver}(\mathsf{crs}, x, \pi) = 1] = 0$$

Perfect Zero-Knowledge: *for all $\mathfrak{K}$, all $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$, all $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$, all $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$ and all $(x, w) \in \mathcal{R}_\rho$, the distributions $\mathsf{Prove}(\mathsf{crs}, x, w)$ and $\mathsf{Sim}(\mathsf{crs}, \mathsf{tk}, x)$ are the same.*

We say that Zero-Knowledge Proofs are Non-interactive (NIZK) if there is no interaction between the prover and the verifier. Note that efficient NIZK proof for pairing equations were found only in 2008 by Groth and Sahai [GS08]. Today their scheme has been optimized and improved in different work like [BFI+10] but this is still the only efficient way to deal with this problem.

2.7 Commitments and Smooth Projective Hash Functions

In this section we will define, give security definition and example of Commitment and Smooth Projective Hash Functions. We will use these two primitives together to achieve different kind of protocols like Oblivious Transfer or Oblivious Language-based Envelop. It has already been used in many cases like Oblivious Signature-based Envelop [BPV12] or Password Authenticated Key Exchange [ABB+13].

Definition 14 (One Way Hash Function Family)**.** *A One Way Hash Function Family is a set $\mathcal{H} = \{H_k\}_{k \in K}$ of functions from a space A to a space B that verifies:*

- *The probability that an adversary find $k \in K, (a, a') \in A^2$ such that $a \neq a'$ and $H_k(a) = H_k(a')$ is negligible. This is called the collision resistance property.*
- *$\forall k \in K$ H_k is efficiently computable.*

2.7.1 Definitions and Security

Commitments allow a user to commit to a value without revealing it, but without the possibility to later change his mind. It is composed of these algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ generates the system parameters param, according to the security parameter $\mathfrak{K}$.
- $\mathsf{KeyGen}(\mathsf{param})$ generates a commitment key ck.
- $\mathsf{Commit}(\mathsf{ck}, m; r)$ produces a commitment c and a opening data δ on the input message $m \in \mathcal{M}$ using the random coins $r \xleftarrow{\$} \mathcal{R}$.

- $\mathsf{VerCom}(\mathsf{ck}, \delta, c, m; r)$ outputs 1 if c is a commitment of m with the randomness r for the commitment key ck along with opening data δ.

Such a commitment scheme should be both **hiding**, which says that the commit phase does not leak any information about m, and **binding**, which says that the decommit phase should not be able to open to two different messages.

Additional features are also sometimes required, such as non-malleability, extractability, and/or equivocability. We may also include a label ℓ[12], which is an additional public information that has to be the same in both the commit and the decommit phases.

A commitment scheme is said **equivocable** if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and two algorithms

- $\mathsf{SimCom}^{\ell}(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
- $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on m.

such that the following properties are satisfied:

- *trapdoor correctness* (all simulated commitments can be opened on any message)
- *setup indistinguishability* (one cannot distinguish the parameters param generated by $\mathsf{SetupCom}$ from the one generated by SimCom)
- *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom) even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said **extractable** if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and a new algorithm

- $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

such that the following properties are satisfied:

[12] The label is here to avoid resend of the exact same commitment, thus the label will contain session's id

- *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$)
- *setup indistinguishability* (as above)
- *binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup in [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09]). A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, onto a set $\mathcal{G}$, is defined by five algorithms $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$ where there exists witness to prove that a word belong to a language;
- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathfrak{L}$;
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, derives the projection key hp from the hashing key hk and the word W.
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $v \in \mathcal{G}$, using the hashing key hk and the word W.
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the hash value $v' \in \mathcal{G}$, using the projection key hp and the witness w that the word $W \in \mathfrak{L}$.

In the following, we assume $\mathfrak{L}$ is a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$. An SPHF should satisfy the following properties:

- *Correctness*: Let $W \in \mathfrak{L}$ and w a witness of this membership. Then, for all param, all hashing keys hk and associated projection keys hp we have
  $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$.
- *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathcal{G} \end{array} \right\}.$$

This is formalized by: $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathcal{G}} |\Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V]|$ is negligible.

- *Pseudo-Randomness*: If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable. For any PPT adversary $\mathcal{A}$, this advantage is negligible:

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]|$$

2.7.2 Examples

Examples of Commitment: As well described in [ABB+13] we have interesting examples of commitment schemes:

An encryption scheme $\mathcal{E}$ can be used as a commitment in the following way:

- $\mathsf{SetupCom}(1^{\mathfrak{K}}) \leftarrow \mathsf{Setup}_{\mathcal{E}}(1^{\mathfrak{K}})$
- $\mathsf{KeyGen}(\mathsf{param}) \leftarrow \mathsf{KeyGen}_{\mathcal{E}}(\mathsf{param})$: $\mathsf{ck} \leftarrow \mathsf{pk}$
- $\mathsf{Commit}(\mathsf{ck}, m) \leftarrow \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{ek}, m; r) : (c, \delta) = (c_{\mathcal{E}}, r)$
- $\mathsf{VerCom}(\mathsf{ck}, \delta, c) \leftarrow (c == \mathsf{Encrypt}_{\mathcal{E}}(\mathsf{ek}, m; r))$

This commitment is perfectly binding since the correctness of the encryption scheme ensure that the message encrypted is m. The scheme is also computationally hiding because of the ind-cpa security of the encryption scheme. Note that from this type of commitment it is easy to have an extractable commitment by adding as extraction algorithm the decryption algorithm of $\mathcal{E}$ the trapdoor will be the decryption key.

The commitment from [AFG+10]:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$: $\mathsf{param} \leftarrow \mathsf{GGen}(1^{\mathfrak{K}})$
- $\mathsf{KeyGen}(\mathsf{param})$: Outputs an independent generator T.
- $\mathsf{Commit}(T, m; r) : (c, \delta) = (g_2^r T^m, g_1^r) \in \mathbb{G}_T$
- $\mathsf{VerCom}(\mathsf{ck}, \delta, c)$: Checks $e(g_1, c/T^m) = e(\delta, g_2) \in \mathbb{G}_T$

It is perfectly hiding: indeed from one commitment every message could have been in it: if $T = g^t$ and $c = g_2^a$ then a valid δ for m is g_1^{a-tm}. The scheme is computationally binding under DDH in $\mathbb{G}_2$. This lead to an equivocable commitment scheme when leaking the discrete logarithm of T as equivocation key.

Chameleon hash functions

Chameleon hash functions can instantiate a perfectly hiding commitment. A chameleon hash function is a hash function where there is an additional algorithm using a trapdoor key, which, given a message a randomness and a second message, can find a second randomness such that the hash of the first message with the first randomness is equal to the hash of the second message with the second randomness. Moreover without the trapdoor key the function should be collision resistant. If a commitment is a hash value since it could be from any message, the commitment is perfectly hiding under the correctness of the chameleon hash function. The commitment is computationaly biding under the collision resistance of the hash function. CDH-based Chameleon Hash [BC15] is a nice example of such scheme and we will use it later.

Definition 15 (Chameleon Hash Function)**.** *A chameleon hash function is made of five algorithm* (KeyGen, VKeyGen, CH, Coll, Valid)

- KeyGen($\mathfrak{K}$) *returns a hashing key* ck *and a trapdoor key* tk*.*
- VKeyGen(ck) *generates the verification keys* vtk *and add an element to* ck
- CH(ck, $m; r$) *returns a hash value of the message* m *using the randomness* r*.*
- Coll(m, r, m', tk) *returns a random* r' *such that the couples* (m, r) *and* (m', r') *have the same hash value.*
- Valid(ck, $m, a, d,$ vtk) *returns 1 if the hash is valid.*

KeyGen($\mathfrak{K}$): Outputs $\mathsf{ck} = (g, h)$ and $\mathsf{tk} = \alpha$, where $g^\alpha = h$ VKeyGen(ck): Appends f to ck and $\mathsf{vtk} = \log_g(f)$	CH(ck, $m; r$): Picks $r \xleftarrow{\$} \mathbb{Z}_p$ outputs $a = h^r g^m$, sets $d = f^r$ Coll(m, r, m', tk): outputs $r' = r + (m - m')/\alpha$ Valid(ck, $m, a, d,$ vtk): Checks $a = h^m \cdot d^{1/\mathsf{vtk}}$

Figure 2.12: CDH-based Chameleon Hash [BC15]

This is a CDH variant of the Pedersen chameleon hash [Ped92] (Figure 2.12).

Example of SPHF: Let's describe a simple example:

Let's say param contains $(g, h) \in \mathcal{G}$

- Language: $\mathfrak{L} = \{(g_1, h_1) | \exists \alpha \in \mathbb{Z}_p, g_1 = g^\alpha \wedge h_1 = h^\alpha\}$, the witness: $w = \alpha$.

- $\mathsf{HashKG}(\mathfrak{L}) \leftarrow \mathsf{hk} = \mathsf{sk}_2 = (x_1, x_2)$
- $\mathsf{ProjKG}(\mathsf{hk}; \mathfrak{L}) \leftarrow \mathsf{hp} = \mathsf{pk}_2 = Y_1^{x_1} g^{x_2}$
- $\mathsf{Hash}(\mathsf{sk}_2; \mathfrak{L}, \mathcal{C}) \leftarrow$
  $e(C_1, \nu)^{x_1} \cdot (e(C_3, g)/(e(h, \mathsf{pk}_1) \cdot e(\nu\mathcal{F}(\mathcal{M}), C_2)))^{x_2}$
- $\mathsf{ProjHash}(\mathsf{pk}_2, \mathfrak{L}, \mathcal{C}, r) \leftarrow e(\mathsf{pk}_2^r, \nu)$

Figure 2.13: SPHF over Linear Encryptions of Valid Waters Signature

- $\mathsf{HashKG}(\mathfrak{L}_M) : \mathsf{hk} \xleftarrow{\$} \mathbb{Z}_p^{k+2}$,
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2))$: Setting $\mathsf{ht} = \begin{pmatrix} \mathsf{pk}_1 \\ \mathsf{pk}_2 + \theta \mathsf{pk}_3 \end{pmatrix}$, we have $[\mathsf{hp}]_2 = \left[\mathsf{hk}^\top \mathsf{ht}\right]_2$, where $\theta = H(\ell, e)$,
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2)) : [H]_2 \leftarrow \left[\mathsf{hk}^\top \left(\mathcal{C} - (0 \mid M)^\top\right)\right]_2$,
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2), \boldsymbol{\mu}) : [H']_2 \leftarrow [\mathsf{hp}\boldsymbol{\mu}]_2$

Figure 2.14: SPHF over $k - \mathsf{MDDH}$ Cramer Shoup Encryption

- Hashing key (private): $\mathsf{hk} = (\lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^2$
- Projection key (public): $\mathsf{hp} = g^\lambda h^\mu$
- Hash: $H = g_1^\lambda h_1^\mu$
- Projection: $H' = \mathsf{hp}^\alpha$

At this point we can make a link between some primitives already mentioned in Figures 2.3 and 2.10. We will describe an SPHF on the language: linear encryptions of valid waters signature which has already been used in the literature [BPV12].

We can build a SPHF over the $k-\mathsf{MDDH}$ Cramer Shoup encryption by following [EHK+13, BBC+13b] described in figure 2.14 ($\mathsf{ek} = (\mathsf{ek}_1, \mathsf{ek}_2, \mathsf{ek}_3)$ coming from the encryption scheme).

2.8 Oblivious Transfer

An Oblivious Transfer is a protocol involving a sender and a receiver that we will sometimes call database and user respectively. It allows a user to have access to a special line in a database that he is requesting. This has to be done under two conditions: first the user should not have any information about the other lines, second the database should not know which line has been requested. Let's describe this formally:

2.8.1 Definition and Security

Definition and Security Model for Oblivious Transfer We will now rigorously describe our OT protocol formalism using languages. It will thus fit what as been said and the formalism of SPHF that we will need.

In this protocol a server $\mathcal{S}$ possesses a database of n lines $(m_1, \ldots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$. And a user U will be able to recover m_k as soon as he requested it. We will say that he is able to recover a line as long as he owns a word $W_k \in \mathfrak{L}_k$[13]. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 16 (Oblivious Transfer)**.** *An* OT *scheme can be formalized in five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DBGen}, \mathsf{Samp}, \mathsf{Verify})$*, along with an interactive protocol* $\mathsf{Protocol}\langle\mathcal{S}, U\rangle$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the number* n*;*

or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[14] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*

- $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all* $i \in \{1, \ldots, n\}$*, description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*
- $\mathsf{Samp}(\mathsf{param}, (\mathsf{sk}_{\mathfrak{L}_i})_{i\in\{1,\ldots,n\}}$[15]$)$ *generates a word* $W_i \in \mathfrak{L}_i$*;*
- $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*
- $\mathsf{Protocol}\langle\mathcal{S}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (m_1, \ldots, m_n)), U((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), W_i)\rangle$*, which is executed between the server* $\mathcal{S}$ *with the private database* $(m_1, \ldots, m_n)$ *and corresponding languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*, and the user* U *with the same languages and the word* W_i*, proceeds as follows. If the algorithm* $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *returns* 1*, then* U *receives* m_k*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

We define the security of Oblivious Transfer in the UC model because we will use only UC-secure Oblivious Transfer.

The ideal functionality of an Oblivious Transfer (OT) protocol was given in [Can01, CKWZ13, ABB+13], and an adaptive version in [GH08]. We describe it in figure 2.15.

[13] The languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ will be assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable, these notion are formally defined in the next section, where we introduce what is an Oblivious-based Language Envelop

[14] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

[15] Optional

The functionality $\mathcal{F}_{(1,n)\text{-}\mathsf{OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1,\ldots,\mathfrak{P}_N$ via the following queries:

- **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ from $\mathfrak{P}_i$**, with $m_k \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathscr{S}$. Ignore further `Send`-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$ from $\mathfrak{P}_j$**, with $s \in \{1,\ldots,n\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathscr{S}$. Ignore further `Receive`-message with the same ssid from $\mathfrak{P}_j$.
- **Upon receiving a message $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ from the adversary** $\mathscr{S}$: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ or $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathfrak{P}_i$ and ignore further `Sent`-message with the same ssid from the adversary.
- **Upon receiving a message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ from the adversary** $\mathscr{S}$: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$ or $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m_s)$ to $\mathfrak{P}_j$ and ignore further `Received`-message with the same ssid from the adversary.

Figure 2.15: Ideal Functionality for 1-out-of-n Oblivious Transfer $\mathcal{F}_{(1,n)\text{-}\mathsf{OT}}$

We will also need the ideal functionality of an adaptive Oblivious Transfer (OT) protocol. It was given in [Can01, CKWZ13, ABB+13], and an adaptive version in [GH08]. We here combine them and rewrite it in simple UC and using our language formalism (this enables us to get rid of `Sent` and `Received` queries from the adversary since the delayed outputs are automatically considered in this simpler framework: We implicitly let the adversary determine if it wants to acknowledge the fact that a message was indeed sent). The first step for the sender (`Send` query) consists in telling the functionality he is willing to take part in the protocol, giving as input his intended receiver and the messages he is willing to send (up to $n_{\max}$ messages). For the receiver, the first step (`Receive` query) consists in giving the functionality the name of the player he intends to receive the messages from, as well as his words. If the word does belong to the language, the receiver recovers the sent message, otherwise, he only gets a special symbol $\perp$. The resulting functionality $\mathcal{F}_{\mathsf{OT}}^{\mathfrak{L}}$ is given in Figure 2.16. Recall that there is no need to give an explicit description of the corruptions in the simple version of UC [CCL15].

The functionality $\mathcal{F}_{\mathsf{OT}}^{\mathfrak{L}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1, \ldots, \mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving from party $\mathfrak{P}_i$ an input $(\texttt{NewDataBase}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$** , with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\texttt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further `NewDataBase`-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, W_k)$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded. Otherwise, reveal $(\texttt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\texttt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m'_k)$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verify}_k(W_k, \mathfrak{L}_k)$ returns 1, and $m'_k = \perp$ otherwise.

(Non-Adaptive case: Ignore further `Receive`-message with the same ssid *from $\mathfrak{P}_j$.)*

Figure 2.16: Ideal Functionality for (Adaptive) Oblivious Transfer $\mathcal{F}_{\mathsf{OT}}^{\mathfrak{L}}$

2.8.2 Example

The example from [ABB+13] is interesting in our context since it makes use of commitments and Smooth Projective Hash Functions. We will talk again about this scheme in Section 5.2.3. The key idea of this scheme is to create a language for each line of the database: for a line m_i we will create the language of all commitments of the number i. Thus the sender will mask each line m_j with the hashed value of the commitment for the language $\{com(j)\}$. But the commitment lie only in one language: it means that the receiver can only recover the hashed value for the language he committed to. Moreover the sender does not know which line has been committed as wished. In the example we use a commitment, a ind-cpa-secure encryption and a Pseudo-Random Generator F with input size equal to the plaintext size and output size equal to the size of the lines in the database.

CRS generation:
$\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow :

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:

1. $\mathcal{R}$ chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. $\mathcal{R}$ computes $(C, \delta_s) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$
3. $\mathcal{R}$ stores δ_s, R, completely erases the random coins used and sends C and c to $\mathcal{S}$

Database answer $(m_1, \ldots, m_n)$:

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and gets $R \leftarrow F(J)$
2. For each line $k = 1, \ldots, n$,
   - $\mathcal{S}$ computes $\mathsf{hk}_k \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_k)$, $\mathsf{hp}_k \leftarrow \mathsf{ProjKG}(\mathsf{hk}_k, \mathfrak{L}_k, (\ell, C))$, $K_k \leftarrow \mathsf{Hash}(\mathsf{hk}_k, \mathfrak{L}_k, (\ell, C))$,
   - $\mathcal{S}$ computes $Q_k \leftarrow R \oplus K_k \oplus m_k$
3. $\mathcal{S}$ erases everything except $(\mathsf{hp}_k, Q_k)_{k=1,\ldots,n}$ and sends it to $\mathcal{R}$.

Data recovery:
Upon receiving $(\mathsf{hp}_k, Q_k)_{k=1,\ldots,n}$, $\mathcal{R}$ computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, \mathfrak{L}_s, (\ell, C), \delta)$ and gets $m_s \leftarrow R \oplus K_s \oplus Q_s$.
Then $\mathcal{R}$ erases everything except m_s.

Figure 2.17: UC-Secure 1-out-of-n OT from an SPHF-Friendly Commitment

Chapter 3

New Featured-Identity-based Encryption and Generalization

First, we present a generalization of IBE named Downgradable IBE that will help us to build a generic transformation from IBE to Attribute-based Encryption in the next chapter. We also describe the transformation to come from this new primitive to other already known Identity-based primitives. Then we present a generic transformation from IBE to Blind IBE. This primitive allows a user to ask for a user secret key without the authority knowing which user has been requested.

3.1 Downgradable Identity-based Encryption

The purpose of this section is not to just add another variant of Identity-based encryption. What we aim here is to create an IBE scheme with the minimal additional properties allowing it to be transformed into Attribute-based Encryption[1]. We found out that this variant of IBE that we named Downgradable IBE can be seen as a generalization of many existing IBE's variant. First, we define this new primitive and then we describe the transformations from it to other Identity-based primitives.

3.1.1 Downgradable Identity-based Encryption

In this section we introduce the notion of Downgradable Identity-Based Encryption. For simplicity we are going to express it in term of Key Encapsulation.

Definition 17 (Downgradable Identity-based Key Encapsulation Scheme)**.** *A Downgradable identity-based key encapsulation (DIBKEM) scheme* DIBKEM *consists of five PPT algorithms* $\mathsf{DIBKEM} = (\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{USKDown})$ *with the following properties.*

[1] We describe the transformation in Section 4.2.

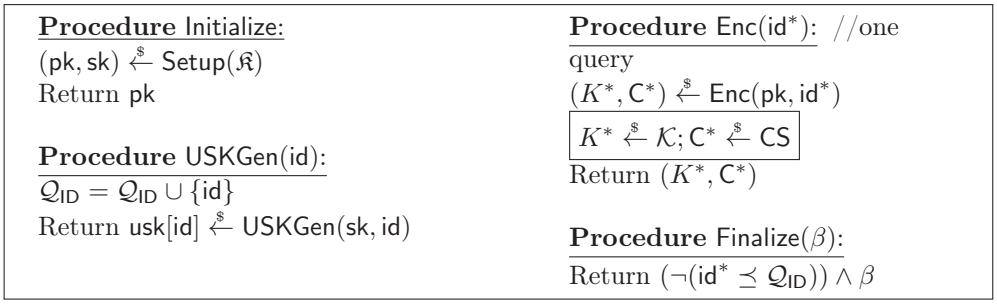


Figure 3.1: Security Games $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{real}}$ and $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}$ (using (K^*,C^*) boxed) for defining PR-ID-CPA-security for DIBKEM.

- *The probabilistic key generation algorithm* $\mathsf{Setup}(\mathfrak{K})$ *returns the (master) public/secret key* $(\mathsf{pk},\mathsf{sk})$. *We assume that* pk *implicitly defines a message space* $\mathcal{M}$*, an identity space* ID*, a key space* $\mathcal{K}$*, and ciphertext space* CS.
- *The probabilistic user secret key generation algorithm* $\mathsf{USKGen}(\mathsf{sk},\mathsf{id})$ *returns the user secret-key* $\mathsf{usk}[\mathsf{id}]$ *for identity* $\mathsf{id} \in \mathsf{ID}$.
- *The probabilistic encapsulation algorithm* $\mathsf{Enc}(\mathsf{pk},\mathsf{id})$ *returns the symmetric key* $K \in \mathcal{K}$ *together with a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to identity* id.
- *The deterministic decapsulation algorithm* $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}],\mathsf{id},\mathsf{C})$ *returns the decapsulated key* $K \in \mathcal{K}$ *or the reject symbol* $\perp$.
- *The probabilistic user secret key downgrade algorithm* $\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}],\tilde{\mathsf{id}})$ *returns the user secret-key* $\mathsf{usk}[\tilde{\mathsf{id}}]$ *as long as* $\tilde{\mathsf{id}} \preceq \mathsf{id}$[2].

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{pk},\mathsf{sk})$ *generated by* $\mathsf{Setup}(\mathfrak{K})$*, all identities* $\mathsf{id} \in \mathsf{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{sk},\mathsf{id})$ *and all* (sk,C) *output by* $\mathsf{Enc}(\mathsf{pk},\mathsf{id})$*:*

$$\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}],\mathsf{id},\mathsf{C}) = \mathsf{sk}] = 1.$$

Moreover, we also require the distribution of $\mathsf{usk}[\tilde{\mathsf{id}}]$ *from* $\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}],\tilde{\mathsf{id}})$ *to be identical to the one from* $\mathsf{USKGen}(\mathsf{sk},\tilde{\mathsf{id}})$.

The security requirements we consider here are indistinguishability and anonymity against chosen plaintext and identity attacks. Instead of defining both security notions separately, we define pseudorandom ciphertexts against chosen plaintext and identity attacks (PR-ID-CPA) which means that challenge key and ciphertext are both pseudorandom. We define PR-ID-CPA-security of DIBKEM formally via the games given in Figure 3.1.

Definition 18 (PR-ID-CPA Security)**.** *A downgradable identity-based key encapsulation scheme* DIBKEM *is* PR-ID-CPA*-secure if for all PPT* $\mathscr{A}$*,*

$\mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBKEM}}(\mathscr{A}) := |\Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{rand}} \Rightarrow 1]|$ *is negligible.*

[2] $\tilde{\mathsf{id}} \preceq \mathsf{id}$ means that for all i, if $\tilde{\mathsf{id}}_i = 1$ then $\mathsf{id}_i = 1$

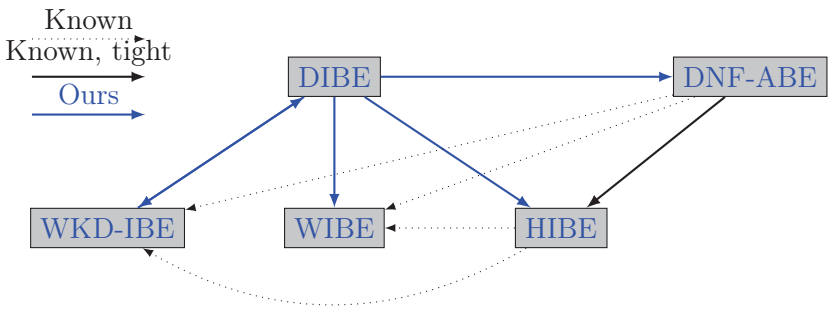


Figure 3.2: Relations Between Primitives

We stress the importance of the condition: $(\neg(\mathsf{id}^* \preceq \mathcal{Q}_{\mathsf{ID}}))$. This is here to guarantee that the adversary did not query an identity that can be downgraded to the challenge one, as this would allow for a trivial attack.

Motivations: The point of creating Downgradable IBE was to construct a primitive the closer from IBE that could be transformed into Attribute-based Encryption. Attribute-Based Encryption (ABE), introduced by Sahai and Waters [SW05], is a generalization of both identity-based encryption and broadcast encryption. It gives a flexible way to define the target group of people who can receive the message: the target set can be defined in a more structural way via access policies on the user's attributes. While broadcast encryption can be obtained from WIBE, as far as we know, this work is the only efficient generic construction of ABE from a variant of IBE. The transformation from DIBE to ABE have for access policies as boolean formulas on the user's attributes in the DNF.

3.1.2 Generalization of Existing Id-based Primitives

As explained previously we will give transformations from DIBE to different variant of IBE. In the figure 3.2 we describe the new transformation we will achieve along with the existing ones.

From DIBE to WIBE

Wildcard Identity-Based Encryption is a concept introduced in [ACD+06]. The idea is to be able to encrypt messages for several identities by fixing some identity bits and letting others free (symbolized by the $*$). Thus only people with identity matching the one used to encrypt can decrypt. We say that id matches id' if $\forall i\ \mathsf{id}_i = \mathsf{id}'_i$ or $\mathsf{id}'_i = *$[3].

Definition 19 (Wildcard Identity-based Key Encapsulation Scheme)**.** *A Wildcard identity-based key encapsulation scheme* WIBKEM *consists of five PPT algorithms* $\mathsf{WIBKEM} = (\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

[3] id_i are bits or group of bits called pattern

- *The probabilistic key generation algorithm* $\mathsf{Setup}(\mathfrak{K})$ *returns the (master) public/secret key* $(\mathsf{pk}, \mathsf{sk})$. *We assume that* pk *implicitly defines a message space* $\mathcal{M}$*, an identity space* $\mathsf{ID} = \{0,1\}^n$*, a key space* $\mathcal{K}$*, and ciphertext space* CS.
- *The probabilistic user secret key generation algorithm* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id})$ *returns the user secret-key* $\mathsf{usk}[\mathsf{id}]$ *for identity* $\mathsf{id} \in \mathsf{ID}$.
- *The probabilistic encapsulation algorithm* $\mathsf{Enc}(\mathsf{pk}, \mathsf{id})$ *returns the symmetric key* $K \in \mathcal{K}$ *together with a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to an identity* $\mathsf{id} \in \hat{\mathsf{ID}} = \{0,1,*\}^n$*, this means that* $\forall i, \mathsf{id}_i \in \{0,1,*\}$.
- *The deterministic decapsulation algorithm* $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \hat{\mathsf{id}}, \mathsf{C})$ *returns the decapsulated key* $K \in \mathcal{K}$ *or the reject symbol* $\perp$.

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{Setup}(\mathfrak{K})$*, all identities* $\mathsf{id} \in \mathsf{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id})$ *and all* $(\mathsf{sk}, \mathsf{C})$ *output by* $\mathsf{Enc}(\mathsf{pk}, \hat{\mathsf{id}})$ *for* $\hat{\mathsf{id}} \in \hat{\mathsf{ID}}$ *such that* $\hat{\mathsf{id}} \preceq_* \mathsf{id}$*:*

$$\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{sk}] = 1.$$

We are now given a $\mathsf{DIBKEM}(\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{USKDown})$, let us show how to build the corresponding WIBKEM.

As with all the following constructions, the heart of the transformation will be to use a DIBKEM for identity of size 2ℓ to handle identities of size ℓ.

Let's consider an identity wid of size ℓ, we define $\mathsf{id} = \phi(\mathsf{wid})$ as follows:

$$\mathsf{id}[2i, 2i+1] = \begin{cases} 01 & \text{if } \mathsf{wid}[i] = 0 \\ 10 & \text{if } \mathsf{wid}[i] = 1 \\ 00 & \text{otherwise.} \end{cases}$$

Now we can define :

- $\mathsf{WIBE.Setup}(\mathfrak{K}) : \mathsf{Setup}(\mathfrak{K})$, except that instead of defining ID as strings of size 2ℓ, we suppose the public key define WID of enriched identities of size ℓ.
- $\mathsf{WIBE.USKGen}(\mathsf{sk}, \mathsf{id}) = \mathsf{USKGen}(\mathsf{sk}, \phi(\mathsf{id}))$.
- $\mathsf{WIBE.Enc}(\mathsf{pk}, \mathsf{id}) = \mathsf{Enc}(\mathsf{pk}, \phi(\mathsf{id}))$.
- $\mathsf{WIBE.Dec}(\mathsf{usk}[\mathsf{id}], \hat{\mathsf{id}}, \mathsf{C})$ checks if $\hat{\mathsf{id}} \preceq \mathsf{id}$, then computes $\mathsf{usk}[\phi(\hat{\mathsf{id}})] = \mathsf{USKDown}(\mathsf{usk}[\phi(\mathsf{id})])$. Returns $\mathsf{Dec}(\mathsf{usk}[\phi(\hat{\mathsf{id}}), \hat{\mathsf{id}}, \mathsf{C})$ or rejects with $\perp$.

The WIBE constructed inherits his security from the DIBE. Due to the construction an adversary against the security of the WIBE will give an adversary against the security of the DIBE.

From DIBE to HIBE

Hierarchical Identity-Based Encryption is a concept introduced in [GS02]. The idea of this primitive is to introduce a hierarchy in the user secret key. A user can create a secret key from his one for any identity with prefix his own identity.

Instead of defining ID as strings of size 2ℓ needed for the transformation, we suppose the public key define HID of enriched identities of size ℓ.

We recall syntax and security of a hierarchical identity-based key encapsulation mechanism (HIBKEM).

Definition 20 (Hierarchical Identity-Based Key Encapsulation Mechanism)**.** *A hierarchical identity-based key encapsulation mechanism* HIBKEM *consists of five PPT algorithms* $\mathsf{HIBKEM} = (\mathsf{Setup}, \mathsf{USKDel}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

- *The probabilistic key generation algorithm* $\mathsf{Setup}(\mathfrak{K})$ *returns the (master) public/secret key and delegation key* $(\mathsf{pk}, \mathsf{sk})$*. We assume that* pk *implicitly defines a message space* $\mathcal{M}$ *and hierarchical identity space* $\mathsf{ID} = \{0,1\}^{\leq n}$*.*
- *The probabilistic user secret key generation algorithm* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id})$ *returns a secret key* $\mathsf{usk}[\mathsf{id}]$ *for hierarchical identity* $\mathsf{id} \in \mathsf{ID}$*.*
- *The probabilistic key delegation algorithm* $\mathsf{USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{id} \in \{0,1\}^p, \mathsf{id}_{p+1} \in \{0,1\}^{\leq n})$ *returns a user secret key* $\mathsf{usk}[\mathsf{id}|\mathsf{id}_{p+1}]$ *for the hierarchical identity* $\mathsf{id}' = \mathsf{id} \mid \mathsf{id}_{p+1} \in \{0,1\}^{p+1}$*. We require* $1 \leq |\mathsf{id}| \leq m-1$*.*
- *The probabilistic encapsulation algorithm* $\mathsf{Enc}(\mathsf{pk}, \mathsf{id})$ *returns a symmetric key* $K \in \mathcal{K}$ *together with a ciphertext* C *with respect to the hierarchical identity* $\mathsf{id} \in \mathsf{ID}$*.*
- *The deterministic decapsulation algorithm* $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$ *returns a decapsulated key* $K \in \mathcal{K}$ *or* $\perp$*.*

For correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{Setup}(\mathfrak{K})$*, all* $\mathsf{id} \in \mathsf{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id})$ *and all* (sk, c) *generated by* $\mathsf{Enc}(\mathsf{pk}, \mathsf{id})$*:*

$$\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{sk}] = 1.$$

Moreover, we also require the distribution of $\mathsf{usk}[\mathsf{id}|\mathsf{id}_{p+1}]$ *from* $\mathsf{USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}], \mathsf{id}, \mathsf{id}_{p+1})$ *to be identical to the one from* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id}|\mathsf{id}_{p+1})$*.*

This time, we are going to map the identity space to a bigger set, with joker identity that can be downgraded to both 0 or 1.

Let's consider an identity hid of size ℓ, we define $\mathsf{id} = \phi(\mathsf{hid})$ as follows:

$$\mathsf{id}[2i, 2i+1] = \begin{cases} 01 & \text{if } \mathsf{hid}[i] = 0 \\ 10 & \text{if } \mathsf{hid}[i] = 1 \\ 11 & \text{otherwise}(\mathsf{hid}[i] = \perp). \end{cases}$$

Now we can define :

- $\mathsf{HIB.Setup}(\mathfrak{K})$: $\mathsf{Setup}(\mathfrak{K})$, except instead of defining ID define HID of enriched identities of size ℓ.

- $\mathsf{HIB.USKGen}(\mathsf{sk}, \mathsf{id}) = \mathsf{USKGen}(\mathsf{sk}, \phi(\mathsf{id}))$. [4]
- $\mathsf{HIB.USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{id} \in \{0,1\}^p, \mathsf{id}_{p+1} \in \{0,1\}) = \mathsf{USKDown}(\mathsf{usk}[\phi(\mathsf{id})], \phi(\mathsf{id}||\mathsf{id}_{p+1}))$. By construction we have $\phi(\mathsf{id}||\mathsf{id}_{p+1}) \preceq \phi(\mathsf{id})$.
- $\mathsf{HIB.Enc}(\mathsf{pk}, \mathsf{id}) = \mathsf{Enc}(\mathsf{pk}, \phi(\mathsf{id}))$.
- $\mathsf{HIB.Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$ returns $\mathsf{Dec}(\mathsf{usk}[\phi(\mathsf{id}), \phi(\mathsf{id}), \mathsf{C})$ or the reject symbol $\perp$.

The HIBE constructed inherits his security from the DIBE. Due to the construction an adversary against the security of the HIBE will give an adversary against the security of the DIBE.

From DIBE to Wicked IBE The paper [AKN07] presents a variant of Identity-based Encryption called Wicked IBE (WKD-IBE). A wicked IBE or wildcard key derivation IBE is a generalization of the concept of limited delegation concept by Boneh-Boyen-Goh [BBG05].

This scheme allows secret key associated with a pattern $P = (P_1, ..., P_l) \in \{\{0,1\}^* \cup \{*\}\}^l$ to be delegated for a pattern $P' = (P'_1, ..., P'_{l'})$ that matches P. We say that P' match P if $\forall i \leq l'$ $P'_i = P_i$ or $P_i = *$ and $\forall l' + 1 \leq i \leq l$ $P_i = *$.

Definition 21 (Wicked Identity-Based Key Encapsulation Mechanism)**.** *A wicked identity-based key encapsulation mechanism* $\mathsf{WKDIB.IBKEM}$ *consists of five PPT algorithms* $\mathsf{WKDIB.HIBKEM} = (\mathsf{WKDIB.Setup}, \mathsf{WKDIB.USKDel}, \mathsf{WKDIB.USKGen}, \mathsf{WKDIB.Enc}, \mathsf{WKDIB.Dec})$ *with the following properties.*

- *The probabilistic key generation algorithm* $\mathsf{WKDIB.Setup}(\mathfrak{K})$ *returns the (master) public/secret key and delegation key* $(\mathsf{pk}, \mathsf{sk})$*. We assume that* pk *implicitly defines a message space* $\mathcal{M}$ *and hierarchical identity space* $\mathsf{ID} = \{0,1\}^{\leq n}$.
- *The probabilistic user secret key generation algorithm* $\mathsf{WKDIB.USKGen}(\mathsf{sk}, \mathsf{id})$ *returns a secret key* $\mathsf{usk}[\mathsf{id}]$ *for hierarchical identity* $\mathsf{id} \in \mathsf{ID}$.
- *The probabilistic key delegation algorithm* $\mathsf{WKDIB.USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{id} \in \{0,1,*\}^n, \mathsf{id}' \in \{0,1\}^n)$ *returns a user secret key* $\mathsf{usk}[\mathsf{id}']$ *for the identity* id' *matching* id[5].
- *The probabilistic encapsulation algorithm* $\mathsf{WKDIB.Enc}(\mathsf{pk}, \mathsf{id})$ *returns a symmetric key* $K \in \mathcal{K}$ *together with a ciphertext* C *with respect to the hierarchical identity* $\mathsf{id} \in \mathsf{ID}$.
- *The deterministic decapsulation algorithm* $\mathsf{WKDIB.Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$ *returns a decapsulated key* $\mathsf{sk} \in \mathcal{K}$ *or* $\perp$.

[4] It should be noted that in case of an HIBKEM, some identities are never to be queried to the downgradable IBKEM: those with 00 is $2i, 2i+1$, or those with 11 at $2i, 2i+1$ and then a 0 (this would correspond to *punctured* identities).

[5] same definition of matching than with WIBE

For correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{WKDIB.Setup}(\mathfrak{K})$*, all* $\mathsf{id} \in \mathsf{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{WKDIB.USKGen}(\mathsf{sk}, \mathsf{id})$ *and all* (sk, c) *generated by* $\mathsf{WKDIB.Enc}(\mathsf{pk}, \mathsf{id})$*:*

$$\Pr[\mathsf{WKDIB.Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{sk}] = 1.$$

Moreover, we also require the distribution of $\mathsf{usk}[\mathsf{id}']$ *from* $\mathsf{USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}], \mathsf{id}, \mathsf{id}_{p+1})$ *to be identical to the one from* $\mathsf{USKGen}(\mathsf{sk}, \mathsf{id}|\mathsf{id}_{p+1})$.

Here again, we are going to map the identity space to a bigger set.

Let us consider an identity id of size ℓ, we define $\mathsf{id} = \phi(\mathsf{wkdid})$ as follows:

$$\mathsf{id}[2i, 2i+1] = \begin{cases} 01 & \text{if } \mathsf{wkdid}[i] = 0 \\ 10 & \text{if } \mathsf{wkdid}[i] = 1 \\ 11 & \text{if } \mathsf{wkdid}[i] = * \end{cases}$$

Now we can define :

- $\mathsf{WKDIB.Setup}(\mathfrak{K})$: $\mathsf{Setup}(\mathfrak{K})$, except instead of defining ID as strings of size 2ℓ, we suppose the public key define WKDID of enriched identities of size ℓ.
- $\mathsf{WKDIB.USKGen}(\mathsf{sk}, \mathsf{id}) = \mathsf{USKGen}(\mathsf{sk}, \phi(\mathsf{id}))$. It should be noted that in case of an WKDIBE, some identities are never to be queried to the downgradable IBE: those with 00.
- $\mathsf{WKDIB.USKDel}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{id}') = \mathsf{USKDown}(\mathsf{usk}[\phi(\mathsf{id})], \phi(\mathsf{id}), \phi(\mathsf{id}'))$.
- $\mathsf{WKDIB.Enc}(\mathsf{pk}, \mathsf{id}) = \mathsf{Enc}(\mathsf{pk}, \phi(\mathsf{id}))$.
- $\mathsf{WKDIB.Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$ returns $\mathsf{Dec}(\mathsf{usk}[\phi(\mathsf{id}), \phi(\mathsf{id}), \mathsf{C})$ or the reject symbol $\perp$.

The Wicked IBE constructed inherits his security from the DIBE. Due to the construction an adversary against the security of the Wicked IBE will give an adversary against the security of the DIBE.

From Wicked IBE to DIBE We can easily transform a Wicked IBE scheme into DIBE by using only identity made of 0 and $*$. In fact the element 1 of the DIBE plays the role of the $*$ of the Wicked IBE. Morally a DIBE can be seen as a Wicked IBE where the patterns are made of only 2 distinct elements instead of 3. We will describe this scheme here since it is not relevant for our purpose. We choose to define a new primitive DIBE because the size of the alphabet for the identities is only 2 and it allows a better efficiency for all schemes based on DIBE. The wildcard $*$ is hard to handle, the best way is how we do it with our DIBE, from that remark it is natural to use DIBE instead of Wicked IBE.

3.1.3 Instantiation and Proof of security

We will base instantiate our DIBE scheme on the IBE scheme from [BKP14]. In their scheme, the identity bits are used in a special way in the user secret

$\underline{\mathsf{Setup}(\mathsf{param})}$:

$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$
For $i = 0, \ldots, \ell$:
$z_i \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times n}; \mathbf{Z}_i = \mathbf{z}_i^\top \cdot \mathbf{A} \in \mathbb{Z}_p^{n\times k}$
$z' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; Z' = {z'}^\top \cdot \mathbf{A} \in \mathbb{Z}_p^{1\times k}$

$\mathsf{pk} := (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0\le i\le \ell}, [Z']_1)$
$\mathsf{sk} := ((z_i)_{0\le i\le \ell}, z')$
Return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{USKGen}(\mathsf{sk}, \mathsf{id} \in \mathsf{ID})}$:

$t \xleftarrow{\$} \mathbb{Z}_p^n$;
$v = \sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id})\mathbf{z}_i t + z' \in \mathbb{Z}_p^{k+1}$
$\mathbf{S} \xleftarrow{\$} \mathbb{Z}_p^{n'\times\mu}$; $\mathbf{T} = \mathbf{B}\cdot\mathbf{S} \in \mathbb{Z}_p^{n\times\mu}$
$\mathbf{V} = \sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id})\mathbf{Z}_i\mathbf{T} \in \mathbb{Z}_p^{(k+1)\times\mu}$
For $i, \mathsf{id}[i] = 1$:
$e_i = \mathbf{Z}_i t \in \mathbb{Z}_p^{k+1}$; $E_i = \mathbf{Z}_i\mathbf{T} \in \mathbb{Z}_p^{k+1\times\mu}$
$\mathsf{usk}[\mathsf{id}] := ([t]_2, [v]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2^{k+1}$
$\mathsf{udk}[\mathsf{id}] := ([\mathbf{T}]_2, [\mathbf{V}]_2, ([e_i]_2, [\mathbf{E}_i]_2)_{i,\mathsf{id}[i]=1})$
$\in \mathbb{G}_2^{n\times\mu} \times \mathbb{G}_2^{(k+1)\times\mu} \times (\mathbb{G}_2^{k+1} \times \mathbb{G}_2^{(k+1)\times\mu})^{\mathsf{Ham}(\mathsf{id})}$
Return $(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}])$

$\underline{\mathsf{Enc}(\mathsf{pk}, \mathsf{id})}$:

$r \xleftarrow{\$} \mathbb{Z}_p^k$
$c_0 = \mathbf{A}r \in \mathbb{Z}_p^{k+1}$
$c_1 = (\sum_{i=0}^{l(\mathsf{id})} f_i(\mathsf{id})\mathbf{Z}_i)\cdot r \in \mathbb{Z}_p^n$
$K = z'_0 \cdot r \in \mathbb{Z}_p$.
Return $\mathsf{sk} = [K]_T$ and $\mathsf{C} = ([c_0]_1, [c_1]_1)$

$\underline{\mathsf{USKDown}(\mathsf{usk}[\mathsf{id}], \tilde{\mathsf{id}})}$:

If $\neg(\tilde{\mathsf{id}} \preceq \mathsf{id})$, then return $\perp$
Set $\mathcal{I} = \{i | \tilde{\mathsf{id}}[i] = 0 \wedge \mathsf{id}[i] = 1\}$
// Downgrading the key:
$\hat{v} = v + \sum_{i\in\mathcal{I}} f_i(\mathsf{id}')e_i \in \mathbb{Z}_p^k$
$\hat{V} = V + \sum_{i\in\mathcal{I}} f_i(\mathsf{id}')E_i \in \mathbb{Z}_p^{k\times\mu}$
// Rerandomization of $(\hat{v}, \hat{V})$:
$s' \xleftarrow{\$} \mathbb{Z}_p^\mu$; $S' \xleftarrow{\$} \mathbb{Z}_p^{\mu\times\mu}$
$t' = t + \mathbf{T}s' \in \mathbb{Z}_p^n$;
$T' = \hat{T}\cdot S' \in \mathbb{Z}_p^{n\times\mu}$
$v' = \hat{v} + \hat{V}\cdot s' \in \mathbb{Z}_p^k$;
$V' = \hat{V}\cdot S' \in \mathbb{Z}_p^{k\times\mu}$
// Rerandomization of e_i:
For $i, \tilde{\mathsf{id}}[i] = 1$:
$e'_i = e_i + \mathbf{E}_i s' \in \mathbb{Z}_p^{k+1}$;
$E'_i = E_i \cdot S' \in \mathbb{Z}_p^{(k+1)\times\mu}$
$\mathsf{usk}[\mathsf{id}'] := ([t']_2, [v']_2)$
$\mathsf{udk}[\mathsf{id}'] := ([T']_2, [V']_2, ([e'_i]_2, [E'_i]_2))$
Return $(\mathsf{usk}[\mathsf{id}'], \mathsf{udk}[\mathsf{id}'])$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$:

Parse $\mathsf{usk}[\mathsf{id}] = ([t]_2, [v]_2)$
Parse $\mathsf{C} = ([c_0]_1, [c_1]_1)$
$\mathsf{sk} = e([c_0]_1, [v]_2)\cdot e([c_1]_1, [t]_2)^{-1}$
Return $\mathsf{sk} \in \mathbb{G}_T$

Figure 3.3: A Downgradable IBE based on MDDH

key: a bits set to 0 does not express in the user secret key. This helps us to create a DIBE by creating the same use secret key and adding for each 1s, an element to eliminate this expression of the 1.

Remark 2. *This instantiation is based on a* MAC *himself based on* HPS. *There is a* MAC *from Naor-Reingold PRF which is tight secure but we did not manage to prove its security when transforming it into a downgradable scheme.*

Theorem 1. *Under the* $\mathcal{D}_k$-MDDH *assumption, the scheme presented in figure 3.3 is* PR-ID-CPA *secure. For all adversaries* $\mathcal{A}$ *there exists an adversary* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and* $\mathbf{Adv}_{\mathsf{DIBKEM},\mathcal{D}_k}(\mathcal{B})^{\mathsf{PR\text{-}ID\text{-}CPA}}(\mathcal{A}) \le (\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{B}) + 2q_k(\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{B}) + 1/q)$ [6].

[6] We recall that q_k is the maximal number of query to the Eval oracle

Initialize:
$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, (x_i)_{0\le i\le \ell}, x'_0) \xleftarrow{\$} \mathsf{Gen}_{\mathsf{MAC}}(\mathsf{par})$
Return $([\mathbf{B}]_2, ([x_i^\top \mathbf{B}]_2)_{0\le i\le \ell})$
 	
Eval(m):
$\mathcal{Q}_{\mathcal{M}} = \mathcal{Q}_{\mathcal{M}} \cup \{\mathsf{m}\}$
$([t]_2, [u]_2) \xleftarrow{\$} \mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{m})$
For $i, \mathsf{m}_i = 1$: $d_i = x_i^\top t \in \mathbb{Z}_p$
Return $([t]_2, [u]_2, ([d_i]_2))$
	Chal(m^*): // one query
	$h \xleftarrow{\$} \mathbb{Z}_p$
	$h_0 = \sum f_i(\mathsf{m}_i^*) x_i \cdot h \in \mathbb{Z}_p^n$
	$h_1 = x'_0 \cdot h \in \mathbb{Z}_p$
	$\boxed{h_1 \xleftarrow{\$} \mathbb{Z}_p}$
	Return $([h]_1, [h_0]_1, [h_1]_T)$
	
	Finalize($\beta \in \{0,1\}$):
	Return $\beta \wedge (\mathsf{m}^* \not\preceq \mathcal{Q}_{\mathcal{M}})$

Figure 3.4: Games $\mathsf{DPR\text{-}CMA}_{\mathsf{real}}$, and $\mathsf{DPR_0\text{-}CMA}_{\mathsf{rand}}$ (boxed) for defining $\mathsf{DPR_0}$-CMA security.

Remark 3. *This instantiation respect the formal definition of* DIBKEM *(definition 17). However for efficiency purpose one can remark that for realizing WIBE or ABE the user's secret keys does not need to be rerandomize during the delegation phase since it will not be used by another user. It introduce the concept of self-delegatable-only scheme. Thus we can avoid the heavy elements T, S, E of the user secret keys. We called this new scheme self-downgradable IBE.*

In this proof we will make use of MAC the primitive and security's definitions can be found in Section 2.4.1.

The inner block is a downgradable MAC

Definition 22. *An affine* MAC *over $\mathbb{Z}_p^n$ is* downgradable*, if the message space is $\mathcal{M} = \{0,1\}^m$ for some finite base set $\{0,1\}$, $f'_0(\mathsf{m}) = 1$, and there exists a public function $f : \mathcal{M} \to \{0, \ldots, \ell\}$ such that for all $\mathsf{m}' \preceq \mathsf{m}$,*

$$f_i(\mathsf{m}'_i) = \begin{cases} f_i(\mathsf{m}_i) & \text{if } \mathsf{m}_i = \mathsf{m}'_i \\ f_i(0) & \text{otherwise} \end{cases}.$$

Let MAC be a delegatable affine MAC over $\mathbb{Z}_p^n$ with message space $\mathcal{M} = \{0,1\}^m$. To build a DIBE, we require a new notion denoted as $\mathsf{DPR_0\text{-}CMA}$ security. It differs from the classical security in two ways. Firstly, additional values needed for DIBE downgrade process are provided to the adversary through the call to Initialize and Eval. Secondly, Chal always returns a real $\mathbf{h}_0$. (In fact, the additional values actually allow the adversary to distinguish real from random $\mathbf{h}_0$.)

Let $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g_1, g_2, e)$ be an asymmetric pairing group in par. Consider the games from Figure 3.4.

Definition 23. *A delegatable affine* MAC *over $\mathbb{Z}_p^n$ is $\mathsf{DPR_0\text{-}CMA}$-secure if for all PPT $\mathscr{A}$, $\mathsf{Adv}^{\mathsf{dpr_0\text{-}cma}}_{\mathsf{MAC}}(\mathscr{A}) := \Pr[\mathsf{DPR\text{-}CMA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{DPR_0\text{-}CMA}^{\mathscr{A}}_{\mathsf{rand}} \Rightarrow 1]$ is negligible.*

$\underline{\mathsf{Gen}_{\mathsf{MAC}}(\mathsf{par})}$:
$\mathbf{B} \xleftarrow{\$} \mathcal{D}_k$
$x_0, \ldots, x_l \xleftarrow{\$} \mathbb{Z}_p^{k+1}; x'_0 \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{sk}_{\mathsf{MAC}} = (\mathbf{B}, x_0, \ldots, x_l, x'_0)$
Return $\mathsf{sk}_{\mathsf{MAC}}$
$\underline{\mathsf{Tag}(\mathsf{sk}_{\mathsf{MAC}}, \mathsf{m})}$:
$s \xleftarrow{\$} \mathbb{Z}_p^k, t = \mathbf{B}s$
$u = (x_0^\top + \sum_{i=1}^{|\mathsf{m}|} \mathsf{m}_i \cdot x_i^\top)t + x'_0 \in \mathbb{Z}_p$
For $i, \mathsf{m}_i = 1$, $d'_i = (-x_i)t$
Return $\tau = ([t]_2, [u]_2, [d]_2) \in \mathbb{G}_2^{k+1} \times \mathbb{G}_2 \times \mathbb{G}_2^{\mathsf{Ham}(\mathsf{m})}$

$\underline{\mathsf{Down}(\tau, \mathsf{m}, \mathsf{m}')}$:
If $\mathsf{m}' \preceq \mathsf{m}$,
$[u']_2 = [u + \sum_{i, \mathsf{m}'_i \neq \mathsf{m}_i} d_i]_2$
$\forall i, \mathsf{m}'_i = 1, [d_i]_2 = [d_i]_2$
Return $\tau' = ([t]_2, [u']_2, [d']_2) \in \mathbb{G}_2^{k+1} \times \mathbb{G}_2 \times \mathbb{G}_2^{\mathsf{Ham}(\mathsf{m}')}$
$\underline{\mathsf{Ver}(\mathsf{sk}_{\mathsf{MAC}}, \tau, \mathsf{m})}$:
If $u = (x_0^\top + \sum_{i=1}^{|\mathsf{m}|} \mathsf{m}_i \cdot x_i^\top)t + x'_0$
then return 1;
Else return 0.

Figure 3.5: Downgradable MAC from HPS [BKP14]

We explicit in Figure 3.5 the inner downgradable MAC we consider in our scheme. And then prove its security.

In this proof we will show that an adversary will be at some point against a standard affine MAC thus the security of the MAC we based our instantiation on, ensure the security of our Downgradable MAC. Intuitively, we will replace query by query the answer of the Eval oracle by pure randomness in $\mathbb{G}_2^{k+1} \times \mathbb{G}_2 \times \mathbb{G}_2^{\mathsf{Ham}(\mathsf{m})}$. This proof is close from the proof of security of the affine MAC from HPS in [BKP14]. G_0 is the real security game defined in 3.4. $\mathsf{G}_{1,i}$ the first $i-1$ answer to the Eval oracle are random and the rest is answered as in the real game. We also need a game to switch from $game_{1,i}$ to the game $\mathsf{G}_{1,i+1}$. This new game will be called $\mathsf{G}'_{1,i}$. Here we will only describe how to come from $\mathsf{G}'_{i,1}$ to $\mathsf{G}_{i+1,1}$ since it is the only part that will differ from the proof in [BKP14].

Let m be the i-th query of the adversary, since $\mathsf{m}^* \not\preceq \mathsf{m}$ there exists a j such that $\mathsf{m}^*_j \neq \mathsf{m}_j$ and $f_j(\mathsf{m}^*_j) \neq f_j(0)$. In this configuration the adversary not more information about x_j than in a standard affine MAC. We can thus reuse the argument of the original proof: there is an information-theoretic argument to show that $u - x'_0$ is uniformly random. To simplify our proof we assume that the adversary $\mathcal{A}$ knows x'_0 and all x_l with $l \notin \{0, j\}$. He may also know $\mathbf{B}^\top x_0$ and $\mathbf{B}^\top x_j$. We will show that $\mathcal{A}$ is unable to guess x_j and x_0, $\mathcal{A}$ has to solve the following matrix equation:

$$\begin{pmatrix} \mathbf{B}^\top x_0 \\ \mathbf{B}^\top \\ h_0 \\ u - x'_0 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathbf{B}^\top & 0 \\ 0 & \mathbf{B}^\top \\ h \cdot \mathbf{I}_{k+1} & \mathsf{m}^*_j h \cdot \mathbf{I}_{k+1} \\ t^\top & m_j t^\top \end{pmatrix}}_{\mathbf{M}} \cdot \begin{pmatrix} x_0 \\ x_j \end{pmatrix} \tag{3.1}$$

The $u - x'_0$ is linearly independent from the other rows: $t^\top$ is independent from $\mathbf{B}^\top$ because $t \notin span(\mathbf{B})$ with probability $(q-1)/q$, also $\mathsf{m}_j \neq \mathsf{m}^*_j$ which means

that this last row is independent from the rows $\begin{pmatrix} h \cdot \mathbf{I}_{k+1} & \mathsf{m}_j^* h \cdot \mathbf{I}_{k+1} \end{pmatrix}$. Thus this system of equations has not enough equations to be solved e.g. $\mathcal{A}$ can not distinguish between a random and u (except for a probability $1/\mathrm{q}$).

Finally, we do all the other steps of the proof like in the original proof, and then we end up with the following lemma.

Lemma 2. *For all adversaries $\mathcal{A}$ there exists an adversary $\mathcal{B}$ with $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ and $\mathbf{Adv}_{\mathsf{MAC}_{\mathsf{HPS}},\mathcal{D}_k}(\mathcal{B})^{\mathsf{DPR}_0\text{-}\mathsf{CMA}}(\mathcal{A}) \leq 2q_k(\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{B}) + 1/q)$.*

Which leads to the security of the downgradable MAC.

Achieving Secure DIBE

We define the sequence of games G_0-G_4 as in Figure 3.6. Let $\mathscr{A}$ be an adversary against the PR-ID-CPA security of DIBKEM. G_0 is the real attack game.

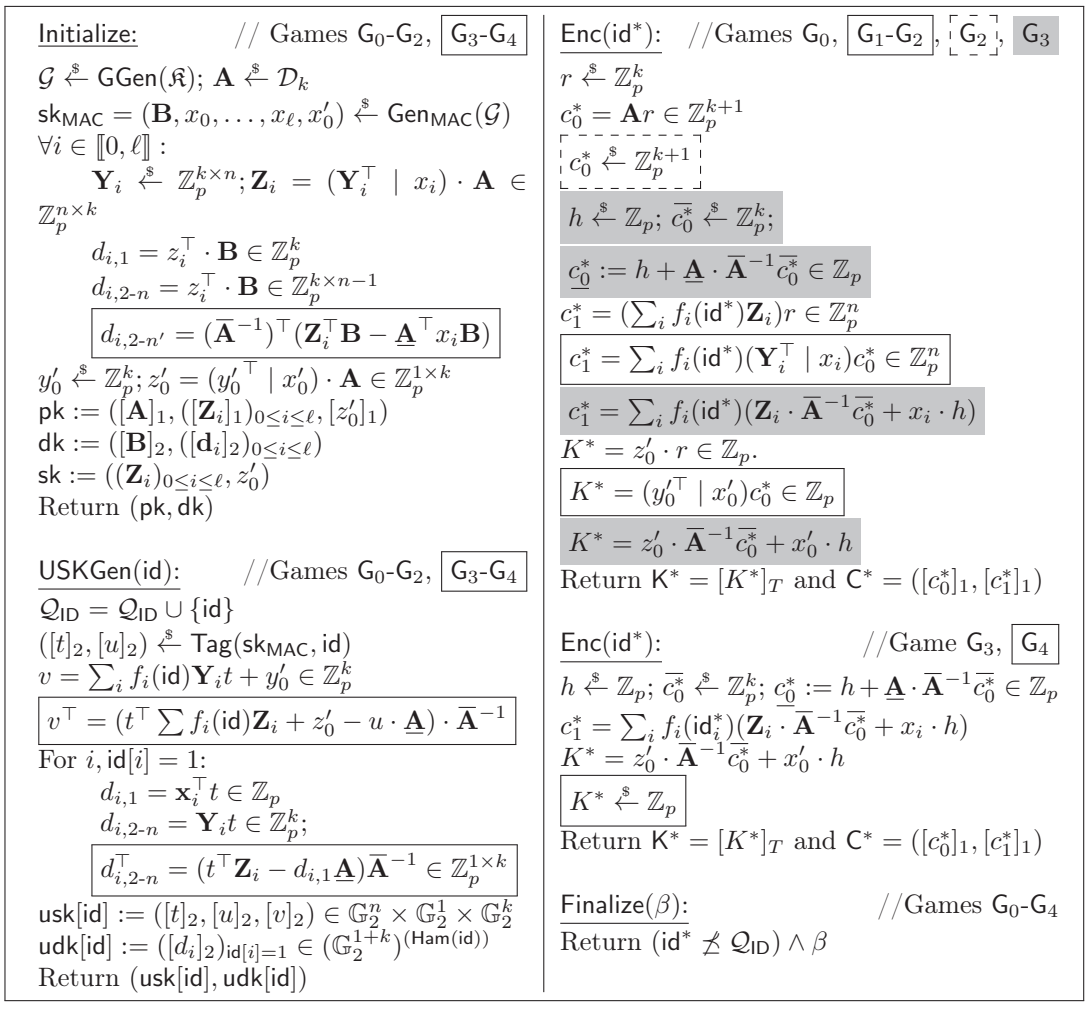


Figure 3.6: Games G_0-G_4 for the proof

We can see that G_1 is simply a rewriting of G_0.

Lemma 3. $\Pr[\mathsf{G}_1^{\mathscr{A}} \Rightarrow 1] = \Pr[\mathsf{G}_0^{\mathscr{A}} \Rightarrow 1]$.

Lemma 4. *There exists an adversary* $\mathcal{B}_1$ *with* $\mathbf{T}(\mathcal{B}_1) \approx \mathbf{T}(\mathscr{A})$ *and* $\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{B}_1) \geq |\Pr[\mathsf{G}_2^{\mathscr{A}} \Rightarrow 1] - \Pr[\mathsf{G}_1^{\mathscr{A}} \Rightarrow 1]|$.

Lemma 5. $\Pr[\mathsf{G}_3^{\mathscr{A}} \Rightarrow 1] = \Pr[\mathsf{G}_2^{\mathscr{A}} \Rightarrow 1]$.

Proof. G_3 is simulated without using y'_0 and $(Y_i)_{0\leq i\leq \ell}$. By $\mathbf{Y}_i^\top = (\mathbf{Z}_i - x_i\underline{\mathbf{A}})\overline{\mathbf{A}}^{-1}$, we have

$$\mathbf{D}_i = (\overline{\mathbf{A}}^{-1})^\top(\mathbf{Z}_i^\top\mathbf{B} - \underline{\mathbf{A}}^\top d_i) = \underbrace{(\overline{\mathbf{A}}^{-1})^\top(\mathbf{Z}_i^\top - \underline{\mathbf{A}}^\top x_i^\top)}_{\mathbf{Y}_i}\mathbf{B}$$
$$d_i = (\overline{\mathbf{A}}^{-1})^\top \cdot (\mathbf{Z}_i^\top t - \underline{\mathbf{A}}^\top \underbrace{x_i^\top t}_{d_i}) = \mathbf{Y}_i t.$$

as in Game G_2. And so, we have $[v]_2$, K^* and C^* are identical to G_2.

□

Lemma 6. *There exists an adversary* $\mathcal{B}_2$ *with* $\mathbf{T}(\mathcal{B}_2) \approx \mathbf{T}(\mathscr{A})$ *and* $\mathsf{Adv}^{\mathsf{dpr}_0\text{-}\mathsf{cma}}_{\mathsf{MAC}}(\mathcal{B}_2) \geq |\Pr[\mathsf{G}_4^{\mathscr{A}} \Rightarrow 1] - \Pr[\mathsf{G}_3^{\mathscr{A}} \Rightarrow 1]|$

Proof. In G_4, we answer the $\mathsf{Enc}(\mathsf{id}^*)$ query by choosing random K^*. We construct algorithm $\mathcal{B}_2$ in Figure 3.7 to show the differences between G_4 and G_3 is bounded by the advantage of breaking $\mathsf{dpr}_0\text{-}\mathsf{cma}$ security of MAC.

We note that, in games G_3 and G_4, the values x_i and x'_i are hidden until the call to $\mathsf{Enc}(\mathsf{id}^*)$ (because the adversary is not allowed to query an id such that $\mathsf{id}^* \preceq \mathsf{id}$). In both games $\mathsf{DPR\text{-}CMA}_{\mathsf{real}}$ and $\mathsf{DPR}_0\text{-}\mathsf{CMA}_{\mathsf{rand}}$, we have $h = \underline{c_0^*} - \underline{\mathbf{A}}\overline{\mathbf{A}}^{-1}\overline{c_0^*}$. Hence $h_0 = \sum f_i(\mathsf{m}_i)x_i \cdot (\underline{c_0^*} - \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1}\overline{c_0^*})$ which implies c_1^* is distributed identically in games G_3 and G_4. If h_1 is uniform (i.e., $\mathcal{B}_2$ is in Game $\mathsf{DPR}_0\text{-}\mathsf{CMA}_{\mathsf{rand}}$) then the view of $\mathscr{A}$ is the same as in G_4. If h_1 is real (i.e., $\mathcal{B}_2$ is in Game $\mathsf{DPR\text{-}CMA}_{\mathsf{real}}$) then $K^* = z'_0 \cdot \overline{\mathbf{A}}^{-1}\overline{c_0^*} + x'_0 \cdot h$, which means the view of $\mathscr{A}$ is the same as in G_3.

□

The proof follows by combining Lemmas 3-6.

In this section we compare the schemes obtained by using our instantiation of DIBE and our new schemes obtained by our transformations and our DIBE. We end up with the most efficient scheme for full security in the standard model and under classical hypothesis for WIBE, WKD-IBE and of similar efficiency for HIBE.

To compare efficiency in a simple way, we choose to consider the case where the number of patterns is maximal e.g. the size of pattern is equal to 1, thus the number of patterns is n which is the length of the identity. The value

Initialize:
$\mathbf{A} \xleftarrow{\$} \mathcal{D}_k$
$([\mathbf{B}]_2, ([x_i^\top \mathbf{B}]_2)_{0\le i\le \ell}) \xleftarrow{\$} \mathsf{Initialize}_{\mathsf{MAC}}$
$\forall i \in [\![0, \ell]\!]$:
$\mathbf{Z}_i \xleftarrow{\$} \mathbb{Z}_p^{n\times k}$; $z'_0 \xleftarrow{\$} \mathbb{Z}_p^{1\times k}$
$\mathsf{pk} := (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0\le i\le \ell}, [z'_0]_1)$
Return $(\mathsf{pk}, \mathsf{dk})$

$\mathsf{Enc}(\mathsf{id}^*)$: //only one query
$([h]_1, [h_0]_1, [h_1]_T) \xleftarrow{\$} \mathsf{Chal}(\mathsf{id}^*)$
$\overline{c_0^*} \xleftarrow{\$} \mathbb{Z}_p^k$; $\underline{c_0^*} := h + \underline{\mathbf{A}} \cdot \overline{\mathbf{A}}^{-1} \overline{c_0^*} \in \mathbb{Z}_p$
$c_1^* = \sum_i f_i(\mathsf{id}^*)\mathbf{Z}_i \cdot \overline{\mathbf{A}}^{-1} \overline{c_0^*} + h_0$
$K^* = z'_0 \cdot \overline{\mathbf{A}}^{-1} \overline{c_0^*} + h_1$
Return $\mathsf{K}^* = [K^*]_T$ and $\mathsf{C}^* = ([c_0^*]_1, [c_1^*]_1)$

$\mathsf{USKGen}(\mathsf{id})$:
$\mathcal{Q}_{\mathsf{ID}} = \mathcal{Q}_{\mathsf{ID}} \cup \{\mathsf{id}\}$
$([t]_2, [u]_2, [T]_2, [u]_2, ([d_i]_2, [D_i]_2)) \xleftarrow{\$} \mathsf{Eval}(\mathsf{id})$
$v^\top = (t^\top \sum f_i(\mathsf{id})\mathbf{Z}_i + z'_0 - u \cdot \underline{\mathbf{A}}) \cdot (\overline{\mathbf{A}})^{-1}$
$V = (\overline{\mathbf{A}}^{-1})^\top (\sum f_i(\mathsf{id}) Z_i^\top \cdot \mathbf{T} - \underline{\mathbf{A}}^\top \cdot u)$
For $i, \mathsf{id}_i = 1$:
$e_i^\top = (t^\top \mathbf{Z}_i - d_i \underline{\mathbf{A}}) \overline{\mathbf{A}}^{-1} \in \mathbb{Z}_p^{1\times k}$
$E_i = (\overline{\mathbf{A}}^{-1})^\top (\mathbf{Z}_i^\top \mathbf{T} - \underline{\mathbf{A}}^\top \cdot D_i) \in \mathbb{Z}_p^{k\times \mu}$
$\mathsf{usk}[\mathsf{id}] := ([t]_2, [u]_2, [v]_2) \in \mathbb{G}_2^n \times \mathbb{G}_2^1 \times \mathbb{G}_2^k$
$\mathsf{udk}[\mathsf{id}] := ([T]_2, [u]_2, [\mathbf{V}]_2, [e_i]_2, [E_i]_2))$
Return $(\mathsf{usk}[\mathsf{id}], \mathsf{udk}[\mathsf{id}])$

$\mathsf{Finalize}(\beta)$:
Return $(\mathsf{id}^* \not\in \mathcal{Q}_{\mathsf{ID}}) \wedge \mathsf{Finalize}_{\mathsf{MAC}}(\beta)$

Figure 3.7: Description of $\mathcal{B}_2$ (having access to the oracles $\mathsf{Initialize}_{\mathsf{MAC}}, \mathsf{Eval}, \mathsf{Chal}, \mathsf{Finalize}_{\mathsf{MAC}}$ for the proof of Lemma 6.

Name	$\lvert\mathsf{pk}\rvert$	$\lvert\mathsf{usk}\rvert$	$\lvert\mathsf{C}\rvert$	assump.	Sec	Loss
WKD [AKN07]	$n+4$	$n+2$	2	BDDH	Sel. standard	$O(Lq_k)$
WKD [AKN07]	$(n+1)n+3$	$n+2$	2	BDDH	Full standard	$O(q_k^L)$
WKD (via our DIBE)	$4n+2$	$3n+5$	5	DLin (any $k-$MDDH)	Full standard	$O(q_k)$
WIBE [BDNS07]	$(n+1)n+3$	$n+1$	$(n+1)n+2$	BDDH	Full standard	$O(L^2 q_k^L)$
WIBE (via our DIBE)	$4n+2$	$3n+5$	5	DLin (any $k-$MDDH)	Full standard	$O(q_k)$

Figure 3.8: Efficiency Comparison Between our Transformations and Previous Schemes

Name	$\|\mathsf{pk}\|$	$\|\mathsf{usk}\|$	$\|\mathsf{C}\|$	assump.	Loss
HIBE [BBG05]	$n+4$	$2+\ell$	5	DLin	sel. $O(n \cdot q_k)$
HIBE [BKP14]	$2n+1$	$11\ell+5$	5	DLin (any $k-$MDDH)	$O(n)$
HIBE (via our DIBE)	$4n+2$	$11n+5$	5	DLin (any $k-$MDDH)	$O(q_k)$

Figure 3.9: Efficiency Comparison Between our Transformations and HIBE schemes

q_k corresponds to the number of derivation key oracle requests made by the adversary[7].

Efficiency comparison for HIBE The Figure 3.9 compares the HIBE built via our DIBE. Our instantiation of DIBE inherits its efficiency from the HIBE from [BKP14], except we need to artificially double the size of the identities. Here ℓ is the number of free bits in an identity (the ones to delegate). Note that for the case of root of the hierarchy e.g. the user with an empty bit string as identity, $\ell = n$.

It should be noted, that while we rely on the same underlying principle, our scheme is not tightly secure and does not rely on the Naor-Reingold PRF, which allows to circumvent the worrisome parts of their proofs[8].

3.2 Blind IBE and Fragmented IBE

In our idea of mixing IBE and SPHF we found another application: efficient Blind fragmented IBE[9] instantiation which will allow to create an efficient adaptive Oblivious Transfer.

3.2.1 Constructing a Blind Fragmented IBKEM from an IBKEM

Shortly speaking a Blind IBKEM is an IBKEM with the possibility to get a user secret key without the authority knowing which identity is related to the request. This feature is in fact very interesting when using IBE as a tool. We will see that in the next chapter 4.3.2.

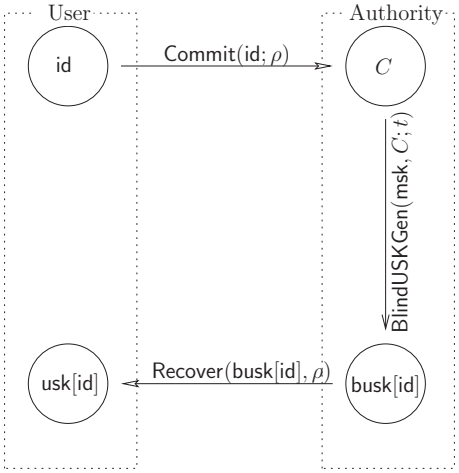
High-Level Idea: achieving Blindness We now show how to obtain a BlindIBKEM scheme from any IBKEM scheme. From a high-level point of view, this transformation mixes two pre-existing approaches.

[7] In the original version of [AKN07] they include an element in the ciphertext to turn their scheme into an encryption scheme. Since our scheme is a Key Encapsulation Mechanism we remove this element when comparing both schemes.

[8] There is problem in the proof which has not been patched at that moment

[9] see Definition 12

First, we are going to consider a reverse Naor transform [BF01, CFH+07] which has been described in Section 2.5. At this point we are seeing IBE keys as signature and we will use the Fischlin [Fis06] round-optimal approach to create blind signatures. The whole interaction is done in one pass: First, the user commits to the message, then he recovers a signature linked to his commitment. For sake of simplicity, instead of using a Non-Interactive Zero-Knowledge Proof of Knowledge of a signature, we are going to follow the [BFPV11, BPV12] approach, where thanks to an additional term, the user can extract a signature on the identity from a signature on the committed identity. This two steps transformation is depicted in Figure 3.10, details will follow in the next paragraph.



1. A user commits to the targeted identity id using some randomness ρ.
2. The authority possesses an algorithm allowing it to generate keys for committed identities using its master secret key msk, and some randomness t, in order to obtain a blinded user secret key busk[id].
3. The user using solely the randomness used in the initial commitment is able to recover the requested secret key from the authority's generated value.

Figure 3.10: Generic Transformation of an IBKEM into a Blind IBKEM (naive approach)

Generic Transformation of an IBKEM into a Blind IBKEM. To blind a user key request we will use a Smooth Projective Hash Function. We assume the existence of a labeled CCA-encryption scheme
$\mathcal{E} = (\mathsf{Setup}_{\text{cca}}, \mathsf{KeyGen}_{\text{cca}}, \mathsf{Encrypt}^{\ell}_{\text{cca}}, \mathsf{Decrypt}^{\ell}_{\text{cca}})$
with an SPHF over a language $\mathfrak{L}^{c}_{\mathsf{id}} \subset X$, where $\mathfrak{L}^{c}_{\mathsf{id}} = \{C \mid \exists \boldsymbol{\rho}$ such that $C = \mathsf{Encrypt}^{\ell}_{\text{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$, defined by $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ onto a set G (where ℓ is a label defined by the global protocol). The formal description of our BlindUSKGen is described in Figure 3.11. Note that we need a key derivation function[10] KDF to get a long enough bit string to hide the user secret key.

Remark 4. *In this transformation the authority sends every secret key but only one can be recovered by the user: the one requested. This comes directly form the property of the* SPHF *and this is why we make use of this primitive. This*

[10] One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string.

- The user computes an encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- For every identity id', the server computes the key $\mathsf{usk}[\mathsf{id}']$ along with a pair of (secret, public) hash keys $(\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'})$ for a smooth projective hash function on the language $\mathfrak{L}^{c}_{\mathsf{id}'}$:

$\mathsf{hk}_{\mathsf{id}'} = \mathsf{HashKG}(\ell, \mathfrak{L}^{c}_{\mathsf{id}'}, \mathsf{param})$ and $\mathsf{hp}_{\mathsf{id}'} = \mathsf{ProjKG}(\mathsf{hk}_{\mathsf{id}'}, \ell, (\mathfrak{L}^{c}_{\mathsf{id}'}, \mathsf{param}), \mathsf{id}')$.

  He also compute the corresponding hash value

$$H_{\mathsf{id}'} = \mathsf{Hash}(\mathsf{hk}_{\mathsf{id}'}, (\mathfrak{L}^{c}_{\mathsf{id}'}, \mathsf{param}), (\ell, C)).$$

  Finally, he sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id', where $\oplus$ is a compatible operation.
- Thanks to $\mathsf{hp}_{\mathsf{id}}$, the user is able to compute the corresponding projected hash value $H'_{\mathsf{id}} = \mathsf{ProjHash}(\mathsf{hp}_{\mathsf{id}}, (\mathfrak{L}^{c}_{\mathsf{id}}, \mathsf{param}), (\ell, C), \boldsymbol{\rho})$. He then recovers $\mathsf{usk}[\mathsf{id}]$ for the initially committed identity id since $H_{\mathsf{id}} = H'_{\mathsf{id}}$.

Figure 3.11: Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a blind IBE

idea of many receiving/one understandable will be reused in this Thesis. We will generalize this with the concept of OLBE *developed in the Section 5.1.*

Theorem 2. *If* IBKEM *is a* PR-ID-CPA*-secure identity-based key encapsulation scheme and $\mathcal{E}$ a labeled* CCA*-encryption scheme compatible with an* SPHF*, then* BlindIBKEM *is leak free and weak blind.*

Proof. First, BlindIBKEM satisfies leak-free secret key generation since it relies on the CCA security on the encryption scheme, forbidding a user to open it to another identity than the one initially encrypted. Furthermore, the pseudo-randomness of the SPHF ensures that the blinded user key received for id is indistinguishable from random if he encrypted $\mathsf{id}' \neq \mathsf{id}$. Finally, the weak blindness also relies on the CCA security on the encryption scheme, since an encryption of id is indistinguishable from a encryption of $\mathsf{id}' \neq \mathsf{id}$. □

Generic Transformation of a Fragmented IBKEM into a Blind Fragmented IBKEM. Now we go a step further and have a look at Fragmented IBE [11]. In a Fragmented IBE a user key is made of pieces from the list $(\mathsf{usk}[0], \mathsf{usk}[0, \mathsf{id}_0], \ldots, \mathsf{usk}[m-1, \mathsf{id}_{m-1}])$ if id_i is the i-th bit of the identity id (for example $\mathsf{usk}[\mathsf{id}] = \mathsf{usk}[0] \cdot \prod_{i=0}^{n} \mathsf{usk}[i, \mathsf{id}_i]$). For Fragmented IBE we are able to improve the efficiency of the Blind Fragmented IBKEM transformation: instead of having a response of the authority linear in the number of user's secret key we will be linear in the number of pieces of key. This new communication cost is logarithmic in the number of user's secret key. We describe this transformation in figure 3.12.

For security reasons, one cannot give directly the evaluation value, but as we are considering the sum of the evaluations for each bit, we simply add a Shamir-like secret sharing, by adding randomness that is going to cancel out at the end.

[11] They were defined in [BKP14]. Affine IBE derive their name from the fact that only affine operations are done on the identity bits (no hashing, square rooting, inverting... are allowed).

- The user computes a bit-per-bit encryption of the requested line identity id and keeps the randomness ρ: $\{C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- The server computes a fragmented version of all the keys $\mathsf{usk}[\mathsf{id}']$, *i.e.* all the values $\mathsf{usk}[i, b]$ for i from 0 up to the length m of the keys and $b \in \{0, 1\}$. He also computes a pair of (secret, public) hash keys $(\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b})$ for a smooth projective hash function on the language $\mathfrak{L}^c_{i,b}$: "The i-th bit of the value encrypted into C is b", *i.e.* $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\ell, \mathfrak{L}^c_{i,b}, \mathsf{param})$ and $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \ell, (\mathfrak{L}^c_{i,b}, \mathsf{param}))$. He also computes the corresponding hash value $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, (\mathfrak{L}^c_{i,b}, \mathsf{param}), (\ell, C))$ and chooses random values z_i. Finally, he sends, for each (i, b), $(\mathsf{hp}_{i,b}, \mathsf{busk}[i, b])$, where $\mathsf{busk}[i, b] = \mathsf{usk}[i, b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i$, together with $Z = \mathsf{usk}_0 \ominus \left(\bigoplus_i z_i\right)$, where $\oplus$ is a compatible operation and $\ominus$ its inverse.
- Thanks to the $\mathsf{hp}_{i,\mathsf{id}_i}$ for the initially committed identity id, the user is able to compute the corresponding projected hash value
$$H'_{i,\mathsf{id}_i} = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, (\mathfrak{L}^c_{i,\mathsf{id}_i}, \mathsf{param}), (\ell, C), \boldsymbol{\rho}),$$
that should be equal to H_{i,id_i} for all i. From the values $\mathsf{busk}[i, \mathsf{id}_i]$, he then recovers $\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i$. Finally, with the operation $\left(\bigoplus_i (\mathsf{usk}[i, \mathsf{id}_i] \oplus z_i)\right) \oplus Z$, he recovers the expected $\mathsf{usk}[\mathsf{id}]$.

Figure 3.12: Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ from a Blind affine IBE

3.2.2 Pairing-Based Instantiation

Affine Bit-Wise Blind IBE. We are going to use the family of IBE from [BKP14] described in the following picture (Figure 3.13), which is their instantiation derived from a Naor-Reingold MAC[12]. In the following, $h_i()$ are injective deterministic public functions mapping a bit to a scalar in $\mathbb{Z}_p$.

We propose a blind user key generation in Figure 3.14.

To fit the framework we will use in the next chapter, we are going to consider the equivocable language of each chameleon hash of the identity bits (a_i, b_{i,m_i}), and then a Cramer-Shoup like encryption of b into d (more details in Figure 2.5). We denote this process as Har in the following protocol, and by $\mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}$ the language on identity bits. And then we will apply the SPHF described in figure 2.14. We thus obtain the following security results.

Theorem 3. *This construction achieves both the weak Blindness, and the leak-free secret key generation requirements under the* $k - \mathsf{MDDH}$ *assumption.*

Proof. Given an adversary $\mathcal{A}$ against the weak Blindness security or the leak-free secret key generation of the blinded scheme, we are going to build an adversary

[12] For the reader familiar with the original result, we combine x, y into a bigger y to lighten the notations, and compact the (x'_i, y'_i) values into a single y' as this has no impact on their construction.

$\underline{\mathsf{Setup}(\mathfrak{K})}$:
$A \xleftarrow{\$} \mathcal{D}_k, B = \overline{A}$
For $i \in [\![0, \ell]\!] : Y_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}; Z_i = Y_i^\top \cdot A \in \mathbb{Z}_p^k$
$y' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; z' = {y'}^\top \cdot A \in \mathbb{Z}_p^k$
$\mathsf{mpk} := (\mathcal{G}, [A]_1, ([Z_i]_1)_{i \in [\![0,\ell]\!]}, [z']_1)$
$\mathsf{msk} := (Y_i)_{i \in [\![0,\ell]\!]}, y'$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})}$:
$s \xleftarrow{\$} \mathbb{Z}_p^k, t = Bs$
$w = (Y_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i) Y_i)t + y' \in \mathbb{Z}_p^{k+1}$
Return $\mathsf{usk}[\mathsf{id}] := ([t]_2, [w]_2) \in \mathbb{G}_2^{k+k+1}$

$\underline{\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})}$:
$r \xleftarrow{\$} \mathbb{Z}_p^k$
$c_0 = Ar \in \mathbb{Z}_p^{k+1}$
$c_1 = (Z_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i) Z_i) \cdot r \in \mathbb{Z}_p$
$K = z' \cdot r \in \mathbb{Z}_p$.
Return $[K]_T$
and $\mathsf{C} = ([c_0]_1, [c_1]_1) \in \mathbb{G}_1^{k+1+1}$

$\underline{\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})}$:
Parse $\mathsf{usk}[\mathsf{id}] = ([t]_2, [w]_2)$
Parse $\mathsf{C} = ([c_0]_1, [c_1]_1)$
$K = e([c_0]_1, [w]_2) \cdot e([c_1]_1, [t]_2)^{-1}$
Return $K \in \mathbb{G}_T$

Figure 3.13: A fragmented IBKEM [BKP14].

- First flow: $\underline{U \text{ starts by computing}}$
  $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_p^{1+4\times\ell}$,
  $a, d = \mathsf{Har}(\mathsf{id}, \ell; \rho) \in \mathbb{Z}_p^{\ell} \times \mathbb{Z}_p^{2\times(k+3)\ell}$,
  Sends $\mathcal{C} = ([a]_1, [d]_2)$ to S
- Second Flow: $\underline{\mathcal{S} \text{ then proceeds}}$
  $s \xleftarrow{\$} \mathbb{Z}_p^k, t = Bs$, $f \xleftarrow{\$} \mathbb{Z}_p^{\ell \times k+1}$,
  For each $i \in [\![1, \lceil \log n \rceil]\!], b \in [\![0, 1]\!]$:
  $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\mathfrak{L}_{\mathsf{Har},i,b}, C)$
  $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, C)$
  $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, C)$
  $\omega_{i,b} = (bY_i)t + f_i + H_{i,b}$

Then sets $w_0 = Y_0 t + y' - \sum_{i=1}^{\ell} f_i \in \mathbb{Z}_p^{k+1}$
Returns $\mathsf{busk} := ([t]_2, [w_0]_2, \{[\boldsymbol{\omega}_{i,b}]_2\}, \{[\mathsf{hp}_{i,b}]_2\})$

- $\mathsf{BlindUSKGen}_3$:
  $\underline{U \text{ then recovers his key}}$
  For each $i \in [\![1, \ell]\!]$:
  $H'_i = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, \mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}, C, \rho_i)$
  $w_i = \omega_{i,\mathsf{id}_i} - H'_i$
  $w = w_0 + \sum_{i=1}^{\ell} w_i$
  And then recovers $\mathsf{usk}[\mathsf{id}] := [t]_2, [w]_2$

Figure 3.14: $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$.

$\mathcal{B}$ against the PR-ID-CPA security of the initial IBE.

From the challenge, $\mathcal{B}$ receives the parameter from an IBE scheme mpk, has access to a user key generation oracle $\mathsf{USKGen}_{\mathcal{O}}$, and for a given fresh id^*, a tuple $(\mathsf{K}^*, \mathsf{C}^*)$ whose consistency he needs to decide.

In the initial game $\mathcal{G}_0$, $\mathcal{B}$ behaves normally, generating $\mathsf{mpk}, \mathsf{msk}$, and answering $\mathcal{A}$ blinded request honestly.

$\mathcal{G}_1$ In this game, $\mathcal{B}$ using the extraction procedure on the CCA commitment, is able to recover the identity id queried by $\mathcal{A}$. If for a bit i, there is two valid openings, then $\mathcal{A}$ broke the collision resistance property of the underlying chameleon hash (and so MDDH) and $\mathcal{B}$ aborts. Thus this game is equivalent to the previous one under $k - \mathsf{MDDH}$.

$\mathcal{G}_2$ In this game, $\mathcal{B}$ starts altering his answer. It is able to recover the identity id queried by $\mathcal{A}$ with the extraction procedure. Now, for each bit of identity there is one binary value not committed $\bar{\mathsf{id}}_i$. For these bits $\mathcal{B}$'s answer will be modified and sets to a random values $\omega_{i,\bar{\mathsf{id}}_i} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. Under the smoothness of the underlying smooth projective hash function, this game is indistinguishable from the previous one.

$\mathcal{G}_3$ Now $\mathcal{B}$ continues to extract the requested id, picks random $f \xleftarrow{\$} \mathbb{Z}_p^{\ell \times k+1}$, and sets $w_0 = \mathsf{usk}[\mathsf{id}] - \sum_{i=1}^{\ell} f_i$, then for $i \in [\![1, n]\!]$, $\omega_{i,\mathsf{id}_i} = f_i + H_{i,\mathsf{id}_i}$ while $\omega_{i,\bar{\mathsf{id}}_i} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ as before. If $\mathcal{B}$, does not recover a valid identity, he simply only sends dummy values.

This game is indistinguishable from the previous one, as this is just a rewriting of the vector f_i. (Noting $\hat{f}_i$ the old one, we have $f_i = \hat{f}_i - (\mathsf{id}_i Y_i)t$ which follows the same distribution).

$\mathcal{G}_4$ $\mathcal{B}$ can now forget about msk, since the user secret key will be whole and hidden in w_0. When receiving a BlindUSKGen request, $\mathcal{B}$ extracts the request identity, and if it is not $\perp$ he forwards it to the $\mathsf{USKGen}_{\mathcal{O}}$ oracle, and plugs the received $\mathsf{usk}[\mathsf{id}]$ as before.

Now $\mathcal{A}$ can request a challenge from $\mathcal{B}$, $\mathcal{B}$ forwards this request to the initial non-blinded IBE challenge for a fresh id^*, and returns the challenge $(\mathsf{K}^*, \mathsf{C}^*)$ to $\mathcal{A}$ which leads to the conclusion. □

For sake of generality, any bit-wise affine IBE could work (like for example Waters IBE [Wat05]). We preferred here a tightly secure IBE giving an even more practical scheme.

Chapter 4

Applications of Identity-based Encryption

In this chapter we will see how Identity-based encryption can be used in different ways. We have already seen that IBE can be transformed into a signature. By adding one more layer of IBE (using HIBE of depth 2) we are able to create Identity-based Signature. This is a known construction [CFH$^+$07] but we will use it in the particular context of constructing Identity-based Designated Verifier Signature (along with other primitives). In a second part we will use our new primitive namely Downgradable Identity-based Encryption and achieve what this primitive has been made for: construct an Attribute-based Encryption scheme from a variant of Identity-based Encryption. At the end of this chapter we describe a construction of adaptive Oblivious Transfer. This technique has already been used but we managed to highly improve the communication cost of this result, achieving a generic construction from any fragmented IBE. We end up with a very efficient and practical adaptive OT (which was not done before).

4.1 Identity Based Designated Verifier Signature

The first section of this chapter is about Designated Verifier Signature (DVS). Our objective is to achieve Identity-based DVS. First we will describe a generic way to construct DVS. In a second time we will show how to move from the normal setup to the Identity-based one. In this first part we will propose a generic way to construct DVS then give an efficient new instantiation.

4.1.1 Definition and Security

The principle of DVS is to allow a signer to sign a message that only one[1] chosen user can verify. We will here describe the scheme as a universal designated verifier signature introduced in [SBWP03].

There are many applications to a such primitive, for example it has the ability to provide a fully deniable email authentication solution where parties can exchange authenticated mail without non repudiation evidence. Thanks to the use of DVS, both parties involved in a mail conversation can authenticate the party they are talking to and are assured that no proof of this conversation can be disclosed. Indeed, if an attacker can gain access to the email inboxes of the trusted parties, he cannot provide a proof of authenticity of the contained information when disclosing emails. ID-based cryptography usage eases the overall process with the reduction of key exchanges needed. Another application is the use of DVS for cloud-based consultation of digitally signed documents. For instance, Electronic Health Records (EHR) and Personnal Health Records (PHR) are now widely deployed in western societies. With this feature, the potential leakage of medical records cannot result in blackmail or insurance fraud as cyber criminals cannot provide a proof of authenticity for the stolen data. Of course, the potential embarrassment caused by the disclosure of medical information without authenticity verification cannot be avoided.

Definition 24. *A Universal Designated Verifier Signature Scheme* DVS *consists of seven PPT algorithms* (Setup, KGS, KGV, Sign, Verify, Des, DesV) *with the following properties:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*: where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param *of the scheme.*
- $\mathsf{KGS}(\mathsf{param})$*: outputs a signing key pair* $(\mathsf{sk}_1, \mathsf{pk}_1)$ *for the signer.*
- $\mathsf{KGV}(\mathsf{param})$*: outputs a verifier key pair* $(\mathsf{sk}_2, \mathsf{pk}_2)$ *for the verifier.*
- $\mathsf{Sign}(\mathsf{sk}_1, \mathcal{M})$*: outputs a signature* σ *on* $\mathcal{M}$ *publicly verifiable with* pk_1*.*
- $\mathsf{Verify}(\mathsf{pk}_1, \mathcal{M}, \sigma)$*: checks the validity of* σ*.*
- $\mathsf{Des}(\sigma, \mathcal{M}, \mathsf{pk}_1, \mathsf{pk}_2)$*: outputs a designated verifier signature* $\hat{\sigma}$ *for the verifier.*
- $\mathsf{DesV}(\hat{\sigma}, \mathcal{M}, \mathsf{sk}_2, \mathsf{pk}_1)$*: checks the validity of the designated signature.*

Using only Setup, KGS, Sign, and Verify we have a simple digital signature scheme.

A Universal Designated Verifier Signature Scheme should satisfy the following properties:

[1] it can easily be extended to a set of multiple users, for example it can be possible to use many times the algorithm Des

- The simple signature scheme $(\mathsf{Setup}, \mathsf{KGS}, \mathsf{Sign}, \mathsf{Verify})$ should be unforgeable (definition at section 2.4.1).

- Only the signer or the designated verifier should be able to generate a verifiable message signature pair for $(\mathcal{M}, \hat{\sigma})$[2]. This is called **DV-Unforgeability** and expressed by the following security experiment ($\mathsf{Exp}^{\mathsf{UDVS}}_{\mathsf{dvuf},\mathcal{A}}$) for an adversary $\mathcal{A}$, where $\mathcal{SM}$ depicts the set of previously signed messages:

$\mathsf{Exp}^{\mathsf{UDVS}}_{\mathsf{dvuf},\mathcal{A}}(\mathfrak{K})$:

$\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$

$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow \mathsf{KGS}(\mathsf{param})$

$(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KGV}(\mathsf{param})$

$(\mathcal{M}, \hat{\sigma}) \longleftarrow \mathcal{A}^{\mathsf{OSign},\mathsf{OVerify}}(\mathsf{pk}_1, \mathsf{sk}_2)$

Return $\mathsf{DesV}(\hat{\sigma}, \mathcal{M}, \mathsf{sk}_2, \mathsf{pk}_1)$ if $\mathcal{M} \notin \mathcal{SM}$

The probability of success against this game is denoted by

$$\mathsf{Succ}^{\mathsf{UDVS}}_{\mathsf{dvuf},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{UDVS}}_{\mathsf{dvuf},\mathcal{A}}(\mathfrak{K}) = 1],$$

- An adversary should not be able to convince a third party about the validity/invalidity of a designated signature, this is called **Non-transferability**[3]. In terms of advantage it means that an adversary must have at best a negligible advantage in distinguishing the two following distributions. We note $\mathcal{S}$ the signature space, and $\xleftarrow{\$}$ to indicate a random uniform sampling:

$$\Delta_0 = \left\{ (\mathcal{M}, \hat{\sigma}) \;\middle|\; \begin{array}{l} (\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KGS}, \mathsf{KGV} \\ \hat{\sigma} = \mathsf{Des}(\mathsf{Sign}(\mathsf{sk}_1, \mathcal{M}), \mathcal{M}, \mathsf{pk}_1, \mathsf{pk}_2) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathcal{M}, \hat{\sigma}) \;\middle|\; \begin{array}{l} (\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathsf{KGS}, \mathsf{KGV} \\ \hat{\sigma} \xleftarrow{\$} \mathcal{S}. \end{array} \right\}$$

Throughout our security experiments, we will use different oracles:

[2] We allow the verifier to be able to create a such signature because if he was not able to then by giving his secret key, he would be able to prove to someone that the signer signed the message.

[3] This is the key property of this primitive

- OSign takes as input $\mathcal{M}$, a signer public key pk_1 and returns a signature σ on this message and adds $\mathcal{M}$ to the set of already signed message $\mathcal{SM}$.
- $\mathsf{OVerify}$ takes as input a designated verifier signature $\hat{\sigma}$, a message $\mathcal{M}$ and a verification key and checks their validity.

Remark 5. *The hard-to-get property is the* ***Non-transferability****. It will force us to use heavy tools to satisfy this property as we describe in the next paragraph. Note that at this point* DVS *and standard digital signature are very different when proving the security: for a standard digital security we can rely on computational problem since the adversary has to produce a value. In the case of the* DVS *the* ***Non-transferability*** *property does not force the adversary to produce anything since it is an indistinguishably between two distributions. It means that we will need to rely the security on a decisional assumption.*

4.1.2 A New Construction

It is natural to start from a signature to build a DVS since the inner scheme should satisfy regular signature's security property (e.g. Unforgeability). This is our starting point, but now we want to transform this signature to have the **Non-transferability** and the **DV-Unforgeability** properties. We will interest in the use of SPHF along with an extractable commitment (an encryption scheme for simplicity). It is an interesting approach because the verifier will have to compute the value (using the SPHF's Hashing algorithm) and verify the equality with the value sent by the signer. Thus the verifier will not be able to convince a third party that he owns a signature on a precise message from the signer because it could have compute it itself. We will detail this in the following.

Starting from the result presented by Blazy et al. [BPV12], we achieve UDVS in 3 steps:

- The signer will sign $\mathcal{M}$ using sk_1 resulting in a signature σ.
- The signer then encrypts σ with an encryption key ek and sends it to the verifier (ek will be a public parameter and no user knows the corresponding private key).
- Both the sender and the verifier will use a SPHF on the language $\mathfrak{L}$ the set composed of encrypted valid signatures from the signer on the message $\mathcal{M}$. In our proposition, the Universal Designated Verifier keys are generated thanks to the SPHF, namely $\mathsf{sk}_2 = \mathsf{hk}$ and $\mathsf{pk}_2 = \mathsf{hp}$. The signature is valid if the verifier receives the same value as the one he computed.

We describe this scheme in the Figure 4.1.

High Level Security Analysis We briefly gives security sketches for the various expected security properties, and show how they reduce in a black-box way to the security of the underlying building blocks.

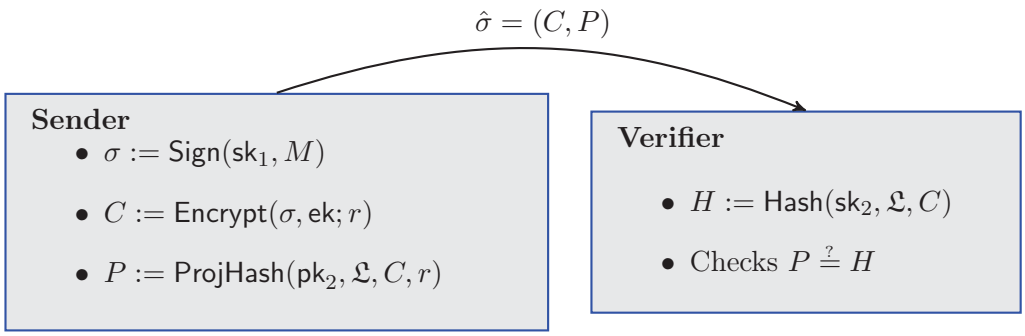


Figure 4.1: Universal Designated Verifier Signature based on SPHF

Theorem 4. *Under the unforgeability of the signature scheme, the proposed Universal Designated Verifier Signature construction is Unforgeable.*

Theorem 5. *Under the unforgeability of the signature scheme, and the Smoothness of the* SPHF*, the proposed Universal Designated Verifier Signature construction is DV-Unforgeable.*

Theorem 6. *Under the indistinguishability (*IND-CPA*) of the encryption scheme and the pseudo randomness of the* SPHF*, the proposed Universal Designated Verifier Signature construction is Non-Transferable.*

Proof of Unforgeability

The inner signing mechanisms of the UDVS is kept unchanged from the underlying signature scheme hence the security remains the same.

Proof of DV-Unforgeability

An adversary $\mathcal{A}$ breaking the DV-unforgeability has a non-negligible probability to produce a valid tuple $(\mathcal{M}, \sigma)$ for a fresh unsigned message (for example verifier).

Proof. We will construct an adversary $\mathcal{B}$ against the unforgeability of the signature scheme. For signatures, $\mathcal{B}$ uses a signing oracle from the initial signature scheme, this is indistinguishable from the real game, and he knows the decryption key sk_1 associated with the encryption key in the common reference string.

When receiving the challenge signature, first $\mathcal{B}$ check if the message is fresh, and the pair valid. In this case, he parses $\hat{\sigma}$ in (C, P). Under the smoothness of the underlying SPHF, C is a valid encryption of a valid signature on M^* for the given public key. Using sk_1, $\mathcal{B}$ can recover a fresh σ^* on M^*, and so breaks the unforgeability of the inner signature scheme. □

Proof of Non-Transferability

An adversary $\mathcal{A}$ breaking the Non-Transferability has a non-negligible probability to distinguish a valid designated signature from an invalid one. We are going to show how to build a simulator $\mathcal{B}$ breaking the IND-CPA security of the underlying encryption scheme.

Proof. For signatures, $\mathcal{B}$ uses a signing oracle from the initial signature scheme, this is indistinguishable from the real game, and he knows the decryption key sk_1 associated with the encryption key in the common reference string.

We are going to proceed in a sequence of games moving from $\mathcal{G}_0$ where the adversary is presented with a valid designated signature, to $\mathcal{G}_2$ where he sees a random value.

- $\mathcal{G}_0$ This is the real game, the signer queries a signature to the signing oracle, encrypts it, and computes honestly the projective hash for the designated identity.
- $\mathcal{G}_1$ The signer now picks a random value for the projected hash, under the pseudo-randomness of the SPHF (implied by definition by the indistinguishability of the encryption), this is indistinguishable from the valid value.
- $\mathcal{G}_2$ The signer now encrypts a random value, once again under the indistinguishability of the encryption, this is indistinguishable from the previous game.

Hence

$$\mathsf{Adv}_{\mathcal{A}}^{NT} \leq 2 * \mathsf{Adv}_{\mathcal{A}}^{\mathsf{ind\text{-}cpa}}$$

□

Instantiation The main difficulty to build this scheme is to find a SPHF on the right language *i.e.* encryptions of a valid signature on $\mathcal{M}$. Here we use the linear encryption over Waters signature [Wat05] followed by the SPHF[4] from [BPV12] . In this instantiation we are able to combine the encryption and the signature in a more efficient scheme by combining these primitives while reusing part of the randomness (see Figure 4.2). Reusing the randomness s allows to have $\sigma_2 = C_2$, this redundancy reduces the overall communication cost by one element without altering the security proof. This optimization is called `Sign&Crypt` leads to a slightly modified version of the language $\mathfrak{L}$ [5]. We want to stress out that this new primitive is different from both the Signcryption introduced by Zheng in [Zhe97] and Signature on Randomizable Ciphertext introduced by Blazy et al. in [BFPV11].

[4] see Figures 2.3 , 2.10 and 2.13 respectively

[5] $\mathfrak{L} := \mathsf{WLin}(\mathsf{ek}, \mathsf{pk}, M)$ denotes the set of elements computed as $\texttt{Sign\&Crypt}(M, \mathsf{ek}, \mathsf{sk})$: this is the set of encryption of valid signature of the message M with the optimization in the writing

$\underline{\mathsf{Setup}(1^{\mathfrak{K}})}$
$\mathcal{G}_g = (p, \mathbb{G}, g, h, \nu, \mathbb{G}_T, e) \leftarrow \mathsf{GGen}(1^{\mathfrak{K}})$
$\mathsf{ek} \leftarrow \text{Lin.}\mathsf{KeyGen}(\mathcal{G}_g)$
$U = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$
$\mathsf{param} = (\mathcal{G}_g, \mathsf{ek}, U)$

$\underline{\mathsf{KGV}(\mathsf{param})}$
$\mathsf{sk}_2 \leftarrow \mathsf{HashKG}(\mathfrak{L}_{\mathsf{ek},\mathsf{pk}_1})$
$\mathsf{pk}_2 \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}_{\mathsf{ek},\mathsf{pk}_1})$
Return $(\mathsf{sk}_2, \mathsf{pk}_2)$

$\underline{\mathsf{KGS}(\mathsf{param})}$
Return
$(\mathsf{sk}_1, \mathsf{pk}_1) := \text{Wat.}\mathsf{KeyGen}(\mathsf{param})$

$\underline{\mathsf{Sign}(\mathcal{M}, \mathsf{sk}_1, \mathsf{pk}_2)}$
$\sigma \leftarrow \text{Wat.}\mathsf{Sign}(\mathsf{sk}_1, \mathcal{M}, s)$
$C \leftarrow \text{Lin.}\mathsf{Encrypt}(\mathsf{ek}, \sigma_1, (r, s))$
$P \leftarrow \mathsf{ProjHash}(\mathsf{pk}_2, \mathfrak{L}, C, r)$
Return (C, P)

$\underline{\mathsf{Verify}(\mathsf{sk}_2, \mathfrak{L}, C, P)}$
$H \leftarrow \mathsf{Hash}(\mathsf{sk}_2, \mathfrak{L}_{\mathsf{ek},\mathsf{pk}_1}, C)$
Return 1 if $H = P$ and 0 otherwise.

Figure 4.2: Instantiation of Sign&Crypt

Additional Properties: In fact, the scheme is more than a mere universal designated verifier signature. By giving the decryption key dk to an authority we allow it to recover a standard signature. Thus, our UDVS scheme is both undeniable and convertible. The undeniability is achieved because the signer can not deny that he signed a message. The convertibility comes from the fact that the authority is able to transform a designated signature into a regular signature.

Security Proof of the SPHF over the Sign&Crypt

Theorem 7. *This Smooth Projective Hash Function (Figure 2.13) is smooth.*

Proof. Let us show that from an information theoretic point of view, $v = \mathsf{Hash}(\mathsf{sk}_2; \mathfrak{L}, \mathcal{C})$ is unpredictable, even knowing hp, when Σ is not a correct Sign&Crypt, in other words $\mathcal{C} = (C_1 = Y_1^r, C_2 = Y_2^{r_2} = \sigma_2, C_3 = \nu^t \sigma_1)$, for $t \neq r_1 + r_2$. (We stress that every incorrect tuple can be written like this)

We recall that we earlier defined $\mathsf{Hash}(\mathsf{hk}, \mathfrak{L}, W) = H := e(C_1, \nu)^{x_1} \cdot (e(C_3, g)/(e(h, \mathsf{pk}_1) \cdot e(\nu \mathcal{F}(\mathcal{M}), C_2)))^{x_2} = e(C_1, \nu)^{x_1} \cdot e(\nu, g^t/C_2)$.

If we denote $Y_1 = g^{y_1}, \nu = g^{y_3}$, and consider the discrete logarithm in $\mathbb{G}_T$ we have:

$$\begin{pmatrix} \log \mathsf{pk}_2 \\ \log H \end{pmatrix} = \begin{pmatrix} y_1 & 1 \\ r_1 y_1 y_3 & (t - r_2) y_3 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

The determinant of this matrix is non-zero iff $\mathcal{C}$ does not belong to the language in other words when $(t \neq r_1 + r_2)$. So H is independent from pk_2 and $\mathcal{C}$. □

4.1.3 Moving To The Identity-based Context

Generic Approach Obviously to move in the identity-based setting we are going to transform the building blocks into an identity-based signature (IBS), and an identity-based SPHF (ID-SPHF). Using identity-based building blocks

we are able to construct IDDVS the same way we constructed DVS, it is describe in Figure 4.3.

$\underline{\mathsf{Setup}(\mathfrak{K})}$ $\mathsf{param} \leftarrow \mathsf{GGen}(\mathfrak{K})$ $\mathsf{mpk}_1, \mathsf{msk}_1, \mathsf{mdk}_1 = \mathsf{HIBKEM.Gen}(\mathfrak{K})$ $\mathsf{mpk}_2, \mathsf{msk}_2 = \mathsf{ID\text{-}SPHF.Setup}(\mathfrak{K})$ $\underline{\mathsf{KGS}(\mathsf{id}_1, \mathsf{msk}_1)}$ Return $\mathsf{sk}_1, \mathsf{dk}_1 := \mathsf{HIBKEM.USKGen}(\mathsf{msk}_1, \mathsf{id}_1)$ $\underline{\mathsf{KGV}(\mathsf{id}_2, \mathsf{msk}_2)}$ Return $\mathsf{sk}_2 := \mathsf{ID\text{-}SPHF.USKGen}(\mathsf{msk}_2, \mathsf{id}_2)$	$\underline{\mathsf{Sign}_d((\mathsf{sk}_1, \mathsf{dk}_1), \mathcal{M}, (\mathsf{mpk}_2, \mathsf{id}_2))}$ $\sigma \leftarrow \mathsf{HIBKEM.USKDel}(\mathsf{mdk}_1, \mathsf{sk}_1,$ $\mathsf{dk}_1, \mathsf{id}_1, \mathcal{M})$ $C, P \leftarrow \mathsf{ID\text{-}SPHF.Encap}(\mathsf{mpk}_2, \mathsf{id}_2; \sigma)$ Return (C, P) $\underline{\mathsf{Verify}(\mathsf{msk}_2, \mathfrak{L}, C, P, \mathsf{id}_1)}$ $H \leftarrow \mathsf{ID\text{-}SPHF.Decap}(\mathsf{msk}_2, C)$ Return 1 if $H = P$ and 0 otherwise

Figure 4.3: ID-DVS from ID-SPHF and HIBKEM

The security will not be different from the scheme in a standard setting:

Theorem 8. *Under the respective security of the* HIBKEM *and the* ID-SPHF*, the Designated Verifier Signature is unforgeable, DV-unforgeable and non-transferable.*

Sketch of Proof: An adversary breaking the (DV)-unforgeability of the ID-DVS either breaks the smoothness of the ID-SPHF (by sending an invalid inner signature with an adapted proof), or the unforgeability of the IBS (by sending a forgery of the inner signature), and so the indistinguishability of the HIBKEM.

Identity-based Signature The difficulty thus lies in the constructions of these Identity-based primitives. We have another constraint, the primitives chosen have to be compatible. For the signature scheme, we use the transformation from a Hierarchical Identity-based Key Encapsulation Mechanism scheme (HIBKEM) to an IBS [CFH+07]. The concept is quite simple, the signature from id_i on $\mathcal{M}$ will be the user secret key $\mathsf{usk}[\mathsf{id}_i||\mathcal{M}]$ derived from $\mathsf{usk}[\mathsf{id}_i]$. Thus the verification is made by decrypting a dummy message encrypted for this new identity.

We supersede the SPHF in the previous approach by an ID-SPHF to achieve ID-DVS described in Figure 4.3. Roughly speaking, the two primitives will be compatible because the SPHF will implicitly verify the signature by computing pairings on the commitment of the signature.

Hierarchical ID-based Key Encapsulation: We use the one presented in [BKP14]. In the present scheme (see Figure 4.4), we only consider a hierarchy of depth 2 which avoids the need to rerandomize the delegation key[6].

Identity-based Designation ID-based Smooth Projective Hash Func-

[6]There is a potential flaw/imprecision in the original proof around the simulation of the randomization of the delegation term, However, this does not arise here as we consider only one delegation step.

$\underline{\mathsf{Gen}(\mathfrak{K})}$:
$\mathbf{A} \xleftarrow{\$} \mathbb{Z}_p^{2\times 1}; \mathbf{B} \xleftarrow{\$} \mathbb{Z}_p$
For $i = 1, \ldots, 2\ell$:
$\mathbf{Y}_i \xleftarrow{\$} \mathbb{Z}_p^2; \mathbf{Z}_i = \mathbf{Y}_i^\top \cdot \mathbf{A} \in \mathbb{Z}_p$
$\mathbf{E}_i = \mathbf{Y}_i \cdot \mathbf{B} \in \mathbb{Z}_p^2$

$y_0' \xleftarrow{\$} \mathbb{Z}_p^2; z_0' = {y_0'}^\top \cdot \mathbf{A} \in \mathbb{Z}_p$
$\mathsf{pk} := (\mathcal{G}, [\mathbf{A}]_1, ([\mathbf{Z}_i]_1)_{0\le i\le \ell}, [z_0']_1)$
$\mathsf{dk} := ([\mathbf{B}]_2, ([\mathbf{E}_i]_2)_{0<i\le 2\ell})$
$\mathsf{sk} := (\mathbf{B}, (\mathbf{Y}_i)_{0\le i\le \ell}, y_0')$
Return $(\mathsf{mpk}_1 = (\mathsf{pk}, \mathsf{dk}), \mathsf{msk}_1 = \mathsf{sk})$

$\underline{\mathsf{USKGen}(\mathsf{msk}_1, \mathsf{id}_1 \in \mathsf{ID})}$:
$t \in \mathbb{Z}_p$
$v = \sum_{i=1}^{\ell} f_i(\mathsf{id}_1)\mathbf{Y}_i t + y_0' \in \mathbb{Z}_p^2$
For $i \in [\![0, \ell-1]\!]$:
$e_i = \mathbf{Y}_{\ell+i} t \in \mathbb{Z}_p^2$
$\mathsf{usk}[\mathsf{id}_1] := ([t]_2, [v]_2) \in \mathbb{G}_2 \times \mathbb{G}_2^2$
$\mathsf{udk}[\mathsf{id}_1] := ([e_i]_2)_{\ell<i\le 2\ell} \in (\mathbb{G}_2^2)^\ell$
Return $\mathsf{sk}_1 = (\mathsf{usk}[\mathsf{id}_1], \mathsf{udk}[\mathsf{id}_1])$

$\underline{\mathsf{USKDel}(\mathsf{dk}, \mathsf{sk}_1, \mathsf{id}_1, \mathcal{M})}$:
$\hat{\mathsf{id}_1} := \mathsf{id}_1 || \mathcal{M}$
$\hat{v} = v + \sum_{i=\ell+1}^{2\ell} f_i(\hat{\mathsf{id}_1}) e_i \in \mathbb{Z}_p^2$
$s' \xleftarrow{\$} \mathbb{Z}_p$, $t' = t + \mathbf{B}s' \in \mathbb{Z}_p$
$v' = \hat{v} + \sum_{i=1}^{2\ell} f_i(\hat{\mathsf{id}_1}) \mathbf{E}_i s' \in \mathbb{Z}_p^2$
$\mathsf{usk}[\mathsf{id}'] := ([t']_2, [v']_2) \in \mathbb{G}_2 \times \mathbb{G}_2^2$
Return $\sigma = \mathsf{usk}[\hat{\mathsf{id}_1}]$

$\underline{\mathsf{Enc}(\mathsf{mpk}_1, \mathsf{id}_1)}$:
$r \xleftarrow{\$} \mathbb{Z}_p$; $c_0 = \mathbf{A}r \in \mathbb{Z}_p^2$
$c_1 = (\sum_{i=1}^{2\ell} f_i(\mathsf{id}_1)\mathbf{Z}_i) \cdot r \in \mathbb{Z}_p$
$K = z_0' \cdot r \in \mathbb{Z}_p$.
Return $\mathsf{K} = [K]_T$ and $\mathsf{C} = ([c_0]_1, [c_1]_1)$

$\underline{\mathsf{Dec}(\mathsf{usk}[\hat{\mathsf{id}_1}], \hat{\mathsf{id}_1}, \mathsf{C})}$:
Parse $\mathsf{usk}[\hat{\mathsf{id}_1}] = ([t]_2, [v]_2)$
Parse $\mathsf{C} = ([c_0]_1, [c_1]_1)$
$\mathsf{K} = e([c_0]_1, [v]_2) \cdot e([c_1]_1, [t]_2)^{-1}$
Return $\mathsf{K} \in \mathbb{G}_T$

Figure 4.4: HIBKEM under SXDH ($k-$MDDH for $k=1$).

tion

We are now going to describe such a compatible ID-based Hash Proof System, which is inspired from the one in the penultimate appendix of [BKP14], except we need some extra tweaking to be compatible with a pairing based verification.

In [BKP14], they presented an ID-Hash proof system based on SXDH. We note that Setup, USKGen, Encap and Decap are the same as in HIBKEM from earlier and Encap* returns a random ciphertext C from the ciphertext space $\mathsf{CS} = \mathbb{G}_1^3$. Correctness of ID-SPHF follows from the correctness of the underlying HIBKEM.

Revisiting the techniques from [BBC$^+$13a], one can extend their construction to handle language more complex like encryption of elements $\alpha \in \mathfrak{L}$. The language corresponding to valid signature is the set of solutions to a pairing product equation which requires some extra care. We need to supersede the USKGen in the ID-SPHF to be compatible with the verification equation of the inner signature. For that, instead of computing t directly in the ID-SPHF the authority receives the values $\mathbf{Z}_i$ from the HIBKEM, and also sends t', a set of $[t\mathbf{Z}_i]_1$.

To sign, a user first computes $\sigma_1 = t_i, \sigma_2 = \sum \mathsf{id}_i' Y_i t_i + y'$. He then uses the ID-SPHF to hide t, for user j, as $c = (\mathbf{G}s, \sum \mathsf{id}_j W_j s + \sum \mathsf{id}_i' \mathbf{Z}_i t_i)$ and $P = [w's]_2$.

User j checks the validity with $[c_0^\top v_j - c_1^\top t_j - P + (\sum \mathsf{id}_i' \mathbf{Z}_i t_i)^\top t_j]_T \stackrel{?}{=} 1_T$, this is done via the ID-SPHF presented in Figure 4.5.

We added some elements compared to the original ID-SPHF but the scheme

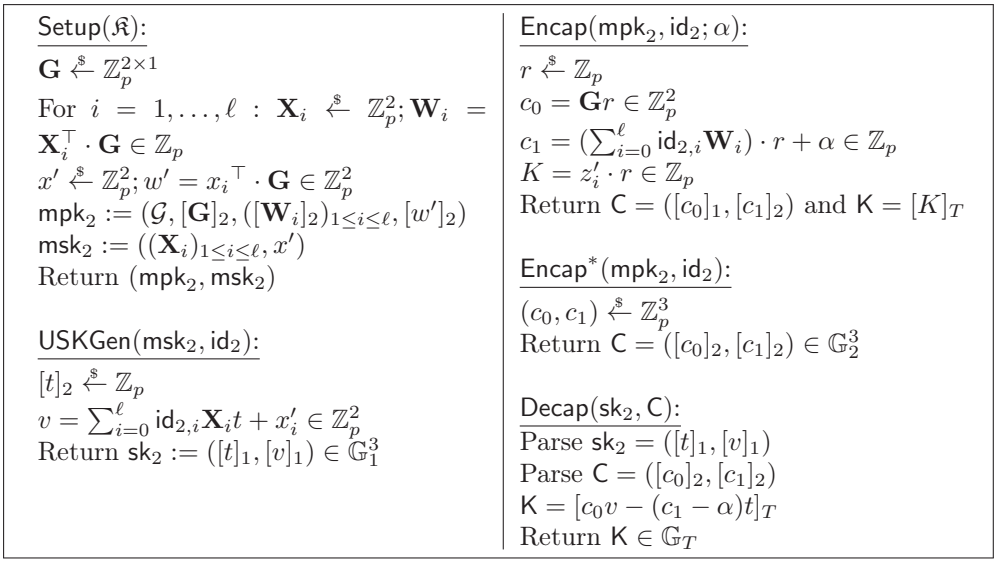


Figure 4.5: Example of an ID-based HPS for the language of encryption.

remains secure since this elements lies in the span of those already given in the original scheme. Under the Self Random Reducibility of the SXDH assumption those extra elements can be simulated.

In terms of communication cost , we manage to keep a lightweight signature made of 5 group elements under SXDH, plus a 128 bits string while we keep a high level security. It is hard to compare with the state of the art for an ID-DVS since they are almost all proven secure in the random oracle model.

Additional properties

The obtained signature in the previous part is a little more than a simple ID-DVS, and reveals to possess various interesting properties. The signature is fully randomizable. So in case a user $\mathcal{A}$ is not able to reach $\mathcal{B}$ directly, he can send something to an intermediate proxy $\mathcal{C}$ which will be unable to learn the validity of the signature. Then $\mathcal{C}$ is able to fully randomize it before sending it to user $\mathcal{B}$ who will be able to learn that it is a signature from $\mathcal{A}$. This would, for instance, allow to hide metadatas in communication preventing competitors to learn that a given enterprise is communicating with a prospective government.

The signature is also an *Undeniable Signature* as the ID-SPHF master authority can revoke the obfuscation of a given signature, and determine whether it is valid or not. It is interesting to note, that the HIBKEM authority has no power at this step, meaning that while the *Judge* has to be unique (e.g. the one with the power of revealing which has signed the message), there can be many sub-authorities in charge of signature (e.g. the HIBE authorities).

4.2 Attribute-based Encryption from Downgradable IBE

Beyond the Notion of Identities, a Step Toward ABE

In an IBE the idea is to describe any user by a bit string. Mail addresses are a classical example. Looking at an address: *Alice@etu.crypto-uni.edu*, one can see that identities englobes some kind of attributes. The user is a student in the university "crypto". From this, it seems natural to extend the construction of IBE to ABE.

This bring us to the link between WIBE and Attribute-based Encryption: using the example of Alice's mail address one can encrypt for every user of the university using the identity "*...**crypto-uni.edu*". The problem with this approach is that WIBE are not practical enough to be used. Moreover an identity representing a lot of attributes could be quite big and thus leads to big ciphertext. Instead we propose to use DIBE and rethink the building of the identity string: now each bit will refer to one attribute. Following our example with Alice, her identity is now a bit string made of 0s for every bits but the ones corresponding to: {university=crypto, rank=student}. The power of this re-conceptualization lies in the construction of the DIBE, in the DIBE a bit set to 0 will not express in the user secret key and thus only the 1s will matter. A user is now completely defined by what he owns and what he does not own.

4.2.1 Definitions and Security

To define Attribute-based Encryption we need to first define what is an access structure[7]:

Definition 25 (Access Structure)**.** *An access structure* $\mathbb{F}$ *is a collection non-empty subsets of the universe of attributes* U*. For simplicity we will index our attributes. Thus we are able to express access structure as follow:* $\mathbb{F} \subseteq 2^U \setminus \{0\}$*.*

Definition 26 (Attribute-based Encryption)**.** *An Attribute-based encryption (ABE) scheme* ABE *consists of four PPT algorithms* $\mathsf{ABKEM} = (\mathsf{Setup}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

- *The probabilistic key generation algorithm* $\mathsf{Setup}(\mathfrak{K})$ *returns the (master) public/secret key* $(\mathsf{pk}, \mathsf{sk})$*. We assume that* pk *implicitly defines a message space* $\mathcal{M}$*, an Attribute space* U*, and ciphertext space* CS*.*
- *The probabilistic user secret key generation algorithm* $\mathsf{USKGen}(\mathsf{sk}, \mathbb{A})$ *that takes as input the master secret key* sk *and a set of attributes* $\mathbb{A} \subset U$ *and returns the user secret-key* $\mathsf{usk}[\mathbb{A}]$*.*
- *The probabilistic encryption algorithm* $\mathsf{Enc}(\mathsf{pk}, \mathbb{F}, M)$ *returns a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to the access structure* $\mathbb{F}$*.*

[7] In our case it will be a boolean formula in Disjunctive Normal From (DNF) it means that the formulae is made of disjunction of conjunction of attribute

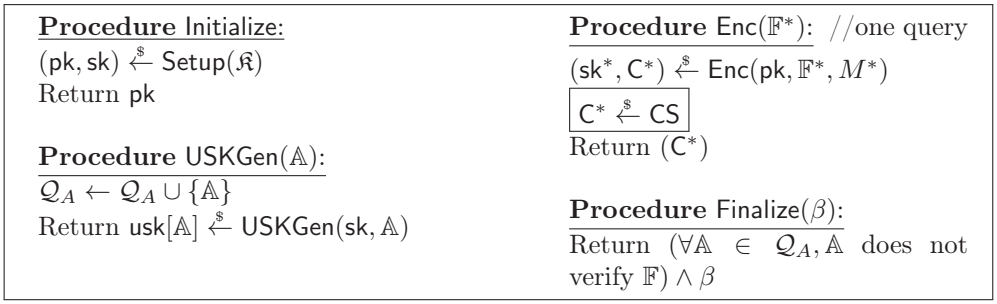


Figure 4.6: Security Games $\mathsf{PR\text{-}A\text{-}CPA}_{\mathsf{real}}$ and $\mathsf{PR\text{-}A\text{-}CPA}_{\mathsf{rand}}$ (boxed) for defining PR-A-CPA-security.

- *The deterministic decryption algorithm* $\mathsf{Dec}(\mathsf{usk}[\mathbb{A}], \mathbb{F}, \mathbb{A}, \mathsf{C})$ *returns the decrypted message* $M \in \mathcal{M}$ *or the reject symbol* $\perp$.

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{Setup}(\mathfrak{K})$*, all access structure* $\mathbb{F}$*, all set of attribute* $\mathbb{A} \subset U$ *satisfying* $\mathbb{F}$*, all* $\mathsf{usk}[\mathbb{A}]$ *generated by* $\mathsf{USKGen}(\mathsf{sk}, \mathbb{A})$ *and all* C *output by* $\mathsf{Enc}(\mathsf{pk}, \mathbb{F}, M)$*:*

$$\Pr[\mathsf{Dec}(\mathsf{usk}[\mathbb{A}], \mathbb{F}, \mathbb{A}, \mathsf{C}) = M] = 1.$$

Like before, we encompass the classical security hypotheses for an ABE, with a PR-A-CPA one as described in Figure 4.6.

Definition 27 (PR-A-CPA Security)**.** *An attribute-based key encapsulation scheme* ABKEM *is* PR-A-CPA*-secure if for all PPT* $\mathscr{A}$*,* $\mathsf{Adv}^{\mathsf{PR\text{-}A\text{-}CPA}}_{\mathsf{ABKEM}}(\mathscr{A}) := |\Pr[\mathsf{PR\text{-}A\text{-}CPA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}A\text{-}CPA}^{\mathscr{A}}_{\mathsf{rand}} \Rightarrow 1]|$ *is negligible.*

Note that we distinguish two types of ABE: the Key Policy ABE (KP-ABE) and the Ciphertext Policy ABE (CP-ABE). The former allows to encrypt for attributes and the user can decrypt if the attributes verify his personal policy lying in his user secret key. The latter is way more practical because the policy lies in the ciphertext, thus the policy can be adapted at every new encryption to target different users. In this thesis we will only talk about CP-ABE.

4.2.2 Transformation from DIBE to ABE

In a usual notion of (ciphertext-policy) ABE, a key is associated with a set $\mathbb{A}$ of attributes in the attribute universe $\mathcal{U}$, while a ciphertext is associated with an access policy $\mathbb{F}$ (or called access structure) over attributes. The decryption can be done if $\mathbb{A}$ satisfies $\mathbb{F}$. We can see that IBE is a special case of ABE where both $\mathbb{A}$ and $\mathbb{F}$ are singletons.

In this thesis, we confine ABE in the two following aspects. First, we restrict the universe $\mathcal{U}$ to be of polynomial size in security parameter; this is often called

small-universe ABE (as opposed to large-universe ABE where $\mathcal{U}$ can be of super polynomial size.). Second, we allow only DNF formulae in expressing policies (as opposed to any boolean formulae, or equivalently, any access structures).

Our idea for obtaining a (small-universe) ABE scheme for DNF formulae from any DIBE scheme is as follows. For simplicity and without lost of generality, we set the universe as $\mathcal{U} = \{1, \ldots, n\}$. We will use DIBE with identity length n. For any set $S \subseteq \mathcal{U}$, we define $\mathsf{id}_S \in \{0,1\}^n$ where its i-th position is defined by

$$\mathsf{id}_S[i] := \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}.$$

To issue an ABE key for a set $\mathbb{A} \subseteq \mathcal{U}$, we use a DIBE key for $\mathsf{id}_{\mathbb{A}}$. On the other hand, to encrypt a message M in ABE with a DNF policy $\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in S_j} a)$, where each attribute a is in $\mathcal{U}$, we encrypt the same message M in DIBE each with id_{S_j} for all $j \in [1,k]$; this will result in k ciphertexts of the DIBE scheme. Note that k is the number of OR, the disjunction, in the DNF formula.

Decryption can be done as follows. Suppose $\mathbb{A}$ satisfies $\mathbb{F}$. Hence, we have that there exists S_j (defined in the formula $\mathbb{F}$) such that $S_j \subseteq \mathbb{A}$. We then derive a DIBE key for id_{S_j} from our ABE key for $\mathbb{A}$ (which is then a DIBE key for $\mathsf{id}_{\mathbb{A}}$); this can be done since $S_j \subseteq \mathbb{A}$ implies that any positions of 1 in id_{S_j} will also contain 1 in $\mathsf{id}_{\mathbb{A}}$ (and thus the derivation is possible). We finally decrypt the ciphertext associated with id_{S_j} to obtain the message M. We summarize this transformation in Figure 4.7.

$\underline{\mathsf{Setup}(\mathsf{param})}$: Run $\mathsf{Setup}_{\mathsf{DIBE}}(\mathfrak{K})$ Return $(\mathsf{pk}, \mathsf{sk})$ $\underline{\mathsf{KeyGen}(\mathsf{sk}, \mathbb{A})}$: Return $\mathsf{usk}[\mathbb{A}] \leftarrow \mathsf{USKGen}_{\mathsf{DIBE}}(\mathsf{sk}, \mathsf{id}_{\mathbb{A}})$	$\underline{\mathsf{Encrypt}(\mathsf{pk}, \mathbb{F}, M)}$: Parse $\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in S_j} a)$ For all $j \in [1,k]$, compute: $(\mathsf{C}_j, K_j) \leftarrow \mathsf{Enc}_{\mathsf{DIBE}}(\mathsf{pk}, \mathsf{id}_{S_j})$ and $C'_j \leftarrow M \oplus K_j$ Return $\mathsf{C} = (\mathsf{C}_1, \ldots, \mathsf{C}_k, C'_1, \ldots C'_k)$ $\underline{\mathsf{Decrypt}(\mathsf{usk}[\mathbb{A}], \mathbb{F}, \mathbb{A}, \mathsf{C})}$: Parse $\mathbb{F} = \bigvee_{j=1}^{k} (\bigwedge_{a \in S_j} a)$ Find $j \in [1,k]$ s.t. $S_j \subseteq \mathbb{A}$ Compute $U \leftarrow \mathsf{USKDown}_{\mathsf{DIBE}}(\mathsf{usk}[\mathbb{A}], \mathsf{id}_{S_j})$ Compute $K_j \leftarrow \mathsf{Dec}_{\mathsf{DIBE}}(U, \mathsf{id}_{S_j}, \mathsf{C}_j)$ Return $M = C'_j \oplus K_j$

Figure 4.7: ABE from DIBE

We have the following security theorem for the above ABE scheme. The proof is very simple and is done by a straightforward hybrid argument over k ciphertexts of DIBE.

Theorem 9. *The above* ABE *from* DIBE *is* PR-A-CPA *secure under the* pr-id-cpa *security of the* DIBE *scheme used. In particular for all adversaries $\mathcal{A}$, we have*

that $\mathsf{Adv}^{\mathsf{PR\text{-}A\text{-}CPA}}_{\mathsf{ABE}}(\mathscr{A}) \leq k \cdot \mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBE}}(\mathscr{A})$ *where* k *is the number of OR in the DNF formula (associated to the challenge ciphertext).*

Proof. We prove our transformation via a sequence of games beginning with the real game for the PR-A-CPA security of the ABE and ending up with a game where the ciphertext of the ABE is uniformly chosen at random e.g. a game where adversary's advantage is reduce to 0.

Let $\mathscr{A}$ be an adversary against the PR-A-CPA security of our transformation. Let C be the simulator of the PR-A-CPA experience.

Game G_0: This is the real security game.

Game $\mathsf{G}_{1.1}$: In this game the simulator generates correctly every ciphertexts but the first one. The first ciphertext is replaced by a random element of the ciphertext space. $\mathsf{G}_{1.1}$ is indistinguishable from G_0 if the pr-id-cpa security holds for the DIBE used.

$$\mathsf{Adv}^{\mathsf{G}_0,\mathsf{G}_{1.1}}(\mathscr{A}) \leq \mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBE}}(\mathscr{A})$$

Game $\mathsf{G}_{1.i}$: This game is the same than the game $\mathsf{G}_{1.i-1}$ but the i-th ciphertext is replaced by a random element of the ciphertext space. $\mathsf{G}_{1.i}$ is indistinguishable from $\mathsf{G}_{1.i-1}$ if the pr-id-cpa security holds for the DIBE used.

$$\mathsf{Adv}^{\mathsf{G}_{1.i-1},\mathsf{G}_{1.i}}(\mathscr{A}) \leq \mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBE}}(\mathscr{A})$$

Game $\mathsf{G}_{1.k}$: in this game all ciphertexts are random elements, $\mathsf{G}_{1.k}$ is indistinguishable from $\mathsf{G}_{1.k-1}$ if the pr-id-cpa security holds for the DIBE used.

$$\mathsf{Adv}^{\mathsf{G}_{1.k-1},\mathsf{G}_{1.k}}(\mathscr{A}) \leq \mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBE}}(\mathscr{A})$$

At this point our current game $\mathsf{G}_{1.k}$ has for challenge encryption only random elements. This means that an adversary has no advantage in winning this game. We finally end up with the advantage of $\mathscr{A}$ in winning the original security game:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{PR\text{-}A\text{-}CPA}}_{\mathsf{ABE}}(\mathscr{A}) &\leq \mathsf{Adv}^{\mathsf{G}_0,\mathsf{G}_{1.k}}(\mathscr{A}) \\
&\leq \sum_{i=1}^{k} \mathsf{Adv}^{\mathsf{G}_{1.i-1},\mathsf{G}_{1.i}}(\mathscr{A}) \\
&\leq k \times \mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{DIBE}}(\mathscr{A})
\end{aligned}$$

□

4.2.3 Efficiency comparison for ABE

Our instantiation leads to a very efficient ABE scheme with tight reduction to a classical primitive. This scheme would be one of the most practical. However

Name	$\|\mathsf{pk}\|$	$\|\mathsf{sk}\|$	$\|\mathsf{C}\|$	pairing	exp $\mathbb{G}$	exp $\mathbb{G}_t$	Reduction Loss
[OT10]	$4U+2$	$3U+3$	$7m+5$	$7m+5$	0	m	$O(q_k)$
[LW12]	$24U+12$	$6U+6$	$6m+6$	$6m+9$	0	m	$O(q_k)$
[CGW15]	$6UR+12$	$3UR+3$	$3m+3$	6	$6m$	0	$O(q_k)$
[Att16] scheme 10	$6UR+12$	$3UR+6$	$3m+6$	9	$6m$	0	$O(q_k)$
[Att16] scheme 13	$96(M+TR)^2+log(UR)$	$3UR+6$	$3m+6$	9	$6m$	0	$O(q_k)$
Our DNF-ABE	$4U+2$	$3U+3$	$3k+2$	13	0	0	$O(k)$

Figure 4.8: Efficiency Comparison of Practical CP-ABE Schemes

we achieve ABE where the access structure has to be a boolean formula in the DNF which is less practical than allowing any kind of access structure (which is done in others practical schemes).

Figure 4.8 presents a non exhaustive comparison of our ABE schemes with efficient ones. They are all fully secure under the classical assumption DLin. U is the size of the universe of attributes. m is the number of attributes in a policy. t is the size of an attribute set, and T is the maximum size of t (if bounded). R is the maximum number of attributes multi used in one policy (if bounded). q_k is again the number of all the key queries made by the adversary during security game. For our scheme, k is the number of OR, the disjunction, in the associated DNF formula.

4.3 From IBE To Oblivious Transfer

4.3.1 Oblivious Transfer Formalism and Main Issues

The Blind IBE (BIBE) we built can instantiate an efficient adaptive Oblivious Transfer, in this section we will see how. In the latest OT in the UC framework [CKWZ13, ABB+13, BC15], the server is required to send an encryption of the complete database for each line required by the user (thus $O(n)$ each time). We here give a protocol requiring $O(\log(n))$ for each line (except the first one, still in $O(n)$), in the UC framework with adaptive corruptions under classical assumptions (MDDH). Our protocol is an adaptive version of the one from [BC15] using an improvement of the ideas from [GH07].

The idea is to encrypt the database with an IBE (each line encrypted for the identity: "number of this line"), and send it once and for all to the recipient. Then the recipient will ask for IBE's private keys in an oblivious way. This second phase can be seen as an oblivious transfer on the database {users' secret keys}. Most of the reduction of the communication cost is done by using IBE with structure on the users' secret keys (Fragmented IBE for example).

Technical issue: In the Blind Fragmented IBKEM transformation, we use implicit decommitment. But in the UC framework it implies a very strong commitment primitive (formalized as SPHF-friendly commitments in [ABB+13]), which is both extractable and equivocable. Our idea is here to split these two properties by using on the one hand an equivocable commitment and on the other hand an (extractable) CCA encryption scheme by generalizing the way to access a line in the database. But this is infeasible with simple line numbers[8]. The change is tiny, we will not consider refer to a number for a target a line of the database (e.g. the identity) but we will use encode them: to request an a line we will commit a word W_i in the language $\mathfrak{L}_i$.

4.3.2 High Level Idea of the Construction:

Our construction builds upon the UC-secure OT scheme from [BC15], with ideas inspired from [GH07], who propose a neat framework allowing to achieve *adaptive* Oblivious Transfer (but not in the UC framework). Their construction is quite simple: It requires a Blind property for the IBE allowing the recovery of the secret key to be oblivious. Note that this method can be seen as realizing an oblivious transfer with a database made of secret keys.

This approach is round-optimal: After the database preparation, the first flow is sent by the user as a commitment to the identity i, and the second one is sent by the server with the blinded expected information. There was several issues to have a UC-secure OT from this construction. We have to handle the possible adaptive corruption request from the adversary in the security games. To do so, as explained before, we use extractable and equivocable commitments using word in languages instead of simple number. Moreover using fragmented IBKEM and blinding them with our transformation we improve the efficiency of our scheme.

Precisely handling adaptive corruptions: To deal with corruptions of the user, recall that a simulated server (knowing the secret key of the encryption scheme) is already able to extract the identity committed to. But we now consider that, for all id, $\mathfrak{L}_{\mathsf{id}}$ is the language of the equivocable commitments on words in the inner language $\widetilde{\mathfrak{L}}_{\mathsf{id}} = \{\mathsf{id}\}$. We assume them to be a *Trapdoor Collection of Languages*, which means that it is computationally hard to sample an element in $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_n$, except for the simulator, who possesses a trapdoor tk (the equivocation trapdoor) allowing it to sample an element in the intersection of languages. This allows a simulated user (knowing this trapdoor) not to really bind to any identity during the commitment phase. The only difference with the algorithm described in Figure 3.12 is that the user now encrypts this word W (which is an equivocable commitment on his identity id) rather than directly encrypting his identity id: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; \boldsymbol{\rho})$.

[8] Thus we have to very slightly change the scheme in figure 3.12

4.3.3 Generic Construction of Adaptive OT

We describe our generic construction of OT in Figure 4.9. We additionally assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. This additional step making use of the encryption scheme is aimed to handle corruptions of any party before the first flow. This leads to the following security result.

Theorem 10. *Assuming that* BlindUSKGen *is constructed as described above, the adaptive Oblivious Transfer protocol described in Figure 4.9 UC-realizes the functionality* $\mathcal{F}_{\mathsf{OT}}^{\mathfrak{L}}$ *presented in Figure 2.16 with adaptive corruptions assuming reliable erasures.*

Proof. We prove the security of this protocol via a sequence of games, starting from the real game, where the adversary $\mathscr{A}$ interacts with the real players, and ending with the ideal game, where we have built a simulator $\mathscr{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathscr{A}$. The simulator is explicitely given in Game 11. Recall that we consider adaptive corruptions.

We denote as $\mathcal{S}$ the server and U the user. The main idea is that, by assumption, the simulator can always obtain the common trapdoor tk of the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ and use it to commit to a word simultaneously belonging to all the languages. In case of adaptive corruption, we face two cases: either the word was not correct, in which case, following the real protocol, the user should have erased his randomness, so that the simulator does not have anything to reveal. Or the word was correct and should belong to a certain language $\mathfrak{L}_s$, and the simulator can then adapt the word and randomness so that it seems to belong to $\mathfrak{L}_s$ (only). This enables us to avoid the use of commitments both extractable and equivocable (which is the usual tool for adaptive corruptions).

Due to the construction of the protocol, we have to prove that the user recovers the secret key $\mathsf{usk}[s]$ corresponding to the s-th line of the database in an oblivious way, which means on the one hand that the user gains no information on the other keys, and on the other hand that the server gains no information on the key required by the user. Assuming this is the case (the proof follows), the adaptive security of the global oblivious transfer relies on the security of the underlying IBE scheme: The indistinguishability of the ciphertexts ensure that the user only recovers the s-th line for which he knows the secret key $\mathsf{usk}[s]$.

Since the channels are authenticated, we know whether a flow was sent by an honest player (and received without any alteration) or not.

Game G_0: This is the real game.

Game G_1: In this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})^a$.
Database Preparation:

1. Server runs $\mathsf{Setup}(\mathfrak{K})$, to obtain $\mathsf{mpk}, \mathsf{msk}$.
2. For each line t, he computes $(D_t, K_t) = \mathsf{Enc}(\mathsf{mpk}, t)$, and $L_t = K_t \oplus DB(t)$.
3. He also computes $\mathsf{usk}[i, b]$ for all $i = 1 \ldots, m$ and $b = 0, 1$ and erases msk.
4. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
5. He then publishes $\mathsf{mpk}, \{(D_t, L_t)\}_t, \mathsf{pk}$.

Index query on s:

1. User chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk:
   $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, S)$
2. User computes $\mathcal{C}$ with the first flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, s, \ell; \boldsymbol{\rho})\rangle)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, U, \mathcal{S})$ (see Figure 3.14).
3. User stores the random $\boldsymbol{\rho}_s = \{\boldsymbol{\rho}_*\}$ needed to open $\mathcal{C}$ to s, and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $(c, \mathcal{C})$ to the Server

IBE input msk:

1. Server decrypts $S \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(S)$
2. Server runs the second flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(U, s, \ell; \boldsymbol{\rho})\rangle)$ on $\mathcal{C}$ (see Figure 3.14).
3. Server erases every new value except $(\mathsf{hp}_{i,b})_{i,b}, (\mathsf{busk}[i, b])_{i,b}, Z \oplus R$ and sends them over a secure channel.

Data recovery:

1. User then using, $\boldsymbol{\rho}_s$ recovers $\mathsf{usk}[s]$ from the values received from the server.
2. He can then recover the expected information with $\mathsf{Dec}(\mathsf{usk}[s], s, D_s) \oplus L_s$ and erases everything else.

[a] using a commitment and a *cpa* secure encryption

Figure 4.9: Adaptive UC-Secure 1-out-of-n OT from a Fragmented Blind IBE

database $(DB(1), \ldots, DB(n))$ and the word W sent by the environment to the server and the user. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, U)$. In case of corruption of an honest player (either server or user), the simulator can give the internal data generated on behalf of the honest users.

Game G_2: In this game, we replace the setup algorithm Setup by SetupT, allowing the existence of a trapdoor to find words in the intersection of the collection of languages. We also allow the simulator to program $\mathsf{Setup}_{\mathsf{CCA}}$ in the CRS, enabling it to learn the extraction trapdoor of the CCA encryption scheme. The indistinguishability of the setups makes this game indistinguishable from the former one for the environment. Corruptions are handled the same way.

Game G_3: We first deal with honest servers $\mathcal{S}$: he computes everything honestly during the database preparation. When receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers s such that $W \in \mathfrak{L}_s$. If it recovers $s \neq t$ such that $W \in \mathfrak{L}_s \cap \mathfrak{L}_t$, then the adversary has broken the infeasibility of finding a word in an intersection of languages without knowing the trapdoor and we abort the game. Otherwise, instead of computing the keys $H_{i,b}$, for $i = 1, \ldots, \ell$ and $b = 0, 1$ with the hash function, the simulator then chooses $H_{i,b} \xleftarrow{\$} G$ when b is not equal to the i-th bit of W.

With an hybrid proof, applying the smoothness for every honest sender, on every index (i, b) such that $b \neq W_i$, since C is extracted to $W \in \mathfrak{L}_i$, for any (i, b) such that $b \neq W_i$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly). This game is thus indistinguishable from the previous one under the smoothness.

Game G_4: Still in this case, when receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers s such that $W \in \mathfrak{L}_s$. Instead of proceeding as the server would do with $(\mathsf{usk}[i, b])$, the simulator proceeds on $(\mathsf{usk}'[i, b])$, with $\mathsf{usk}'[i, b] = 0$ except if $b = W_i$. Since the masks $H_{i,b}$, for $b \neq W_i$, are random, this game is perfectly indistinguishable from the previous one.

Game G_5: We now deal with honest users U: the simulator now uses the trapdoor tk to find a word W' in the intersection of all languages.

With an hybrid proof, applying a security game in each session in which the simulator does not know the trapdoor tk, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value W, thus s.

Game G_6: We deal with **the generation of R for honest servers $\mathcal{S}$ where the users U are honests**: if $\mathcal{S}$ and U are honest at least until $\mathcal{S}$

received the second flow, the simulator sets $R = F(S')$ for both $\mathcal{S}$ and U, with S' a random value, instead of $R = F(S)$.

With an hybrid proof, applying the IND-CPA property for each session, one can show the indistinguishability of this game with the previous one.

Game G_7: Still in the same case, the simulator sets R as a random value, instead of $R = F(S')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game G_8: We now deal with **the generation of H_{i,W_i} for honest servers $\mathcal{S}$ with honest users U**. Thanks to the additional random mask R, one can send random $(\mathsf{usk}[i, W_i])_i$, and H_{i,W_i} can be computed later (when U actually receives its flow).

As above, but only if U has not been corrupted before receiving its flow, the simulator chooses $H_{i,W_i} \xleftarrow{\$} G$. With an hybrid proof, applying the pseudo-randomness, for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C. If the player U involved in the pseudo-randomness game gets corrupted (but the decommitment information is unknown) we are not in this case, and we can thus abort it.

In case of corruption of $\mathcal{S}$, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly).

In case of corruption of the receiver U, and thus receiving the value $\widetilde{DB(s)}$, the simulator computes $\tilde{K}_s$ such that $\widetilde{K_s} \oplus \widetilde{DB(s)} = K_s \oplus DB(s)$ and the corresponding $\widetilde{\mathsf{usk}[W]}$. It chooses R (because it was a random value unknown to the adversary and all the other $H_{i,b}$ are independent random values too) such that $\left(\bigoplus_i \mathsf{busk}[i, W_i] \oplus H'_{i,W_i} \right) \oplus Z \oplus R = \widetilde{\mathsf{usk}[W]}$.

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Remark. We now explain how, in the pairing instantiation which will follow, given already sent values $D_s, K_s \oplus DB(s)$, a simulator recovering from the environment the value $\widetilde{DB(s)}$, can adaptively be able to change his memory so as to compute a user key $\widetilde{\mathsf{usk}[W]}$ such that $\mathsf{Dec}(\widetilde{\mathsf{usk}[W]}, W, D_s) = DB(s) \oplus K_s \oplus \widetilde{DB(s)}$. This is exactly where we use the restriction on the size of DB elements so that we can manage to find a vector $\delta_s \in \mathbb{G}_2^{2k+1}$ such that $DB(s) \oplus \widetilde{DB(s)} = e(D_s, \delta_s)$. Thus, this allows the server to update his memory into $\widetilde{\mathsf{usk}[W]} = \mathsf{usk}[W] \cdot \delta_s$.

Game G_9: Still in this case, the simulator proceeds on $(\mathsf{usk}[i, b])$, with $\mathsf{usk}[i, b] = 0$ for all i, b. Since the masks $H_{i,b}$, $Z \oplus R$, for any i, b, are independent random values (the $\mathsf{busk}[i, b]$, for $b \neq W_i$ are independent random values, and

R is independently random), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index s given by the ideal functionality to the honest receiver U, to simulate $\mathcal{S}$ (but it is still necessary to simulate U).

Game G_{10}: We do not use anymore the knowledge of s when simulating an **honest user** U: the simulator generates a word W' in the intersection of the languages and $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W'; \boldsymbol{\rho})$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$, to send C during the first phase of honest users. This does not change anything from the previous game since the randomness needed to open to a word in another language is never revealed.

When it thereafter receives $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{hp}_{i,b}, \mathsf{busk}[i,b]))$ from the adversary, the simulator computes, for all lines t, $\mathsf{usk}[W_t]$ and recovers K_t and finally $DB(t)$, which provides the database $(DB(1), \ldots, DB(n))$ submitted by the sender. It uses them to send a `Send`-message to the ideal functionality.

Game G_{11}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionality, which means that an honest server has sent a pre-flow and a database, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent a first flow, the simulator generates a word W' in the intersection of languages, $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; \boldsymbol{\rho})$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$ and $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$ where S is a random value;
- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value W and recovers s (aborting in case of multiple values), and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as S, and computes $R = F(S)$);
- when receiving $(\mathsf{hp}_{i,b}, \mathsf{busk}[i,b])$ from the adversary, the simulator computes, for all lines t, $\mathsf{usk}[W_t]$ and recovers K_t and finally $DB(t)$, which provides the database $(DB(1), \ldots, DB(n))$ submitted by the sender. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with $\widetilde{DB(s)}$, on behalf of a corrupted receiver, from the extracted W leading to s, instead of proceeding as the sender would do on $(\mathsf{usk}[i,b])$, the simulator proceeds on $(\mathsf{usk}'_{[}i,b])$, with $\mathsf{usk}'_{[}i,b] = 0$ except if $b = W_i$.

- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(\mathsf{usk}'[i,b])$, with $\mathsf{usk}'[i,b] = 0$ except if $b = W_i$, but it chooses R uniformly at random instead of choosing it as $R = F(S)$; in case of corruption afterward, the simulator will adapt R such that $\left(\bigoplus_i \mathsf{busk}[i, W_i] \oplus H'_{i,W_i}\right) \oplus Z \oplus R = \widetilde{\mathsf{usk}[W]}$, where $\widetilde{\mathsf{usk}[W]}$ leads to $\tilde{K}_s$ such that $\widetilde{K_s} \oplus \widetilde{DB(s)} = K_s \oplus DB(s)$, where $\widetilde{DB(s)}$ is the message actually received by the receiver.

Any corruption either reveals s earlier, which allows a correct simulation of the receiver, or reveals $(DB(1), \ldots, DB(n))$ earlier, which allows a correct simulation of the server. When the server has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the user is corrupted after the server has sent is flow, but before the user receives it, since he has kept $\boldsymbol{\rho}$: this would enable the adversary to recover $\mathsf{usk}[W]$ from $\mathsf{busk}[i, W_i]$ and hp_{i,W_i}. This is the goal of the ephemeral mask R that provides a secure channel.

□

Adaptive UC-Secure Oblivious Transfer. Using the instantiations of the fragmented IBE in figure 3.13 and the transformation 3.14, we finally get our instantiation by combining this $k - \mathsf{MDDH}$-based blind IBE with a $k - \mathsf{MDDH}$ variant of El Gamal for the CPA encryption needed.

The requirement on the IBE blind user secret key generation (being able to adapt the key if the line changes) is achieved assuming that the server knows the discrete logarithms of the database lines[9]. For practical applications, one could imagine to split all 256-bit lines into 8 pieces for a decent/constant trade-off in favor of computational efficiency.

Efficiency: for $k = 1$, so under the classical SXDH assumption, the first flow requires $8 \log |DB|$ elements in $\mathbb{G}_1$ for the CCA encryption part and $\log(|DB| + 1)$ in $\mathbb{G}_2$ for the chameleon one, while the second flow would now require $1 + 4 \log |DB|$ elements in $\mathbb{G}_1$, $1 + 2 \log |DB|$ for the fragmented masked key, and $2 \log |DB|$ for the projection keys. As said before other OT scheme are under non classical security assumption [GH07] or have a communication cost linear in the size of the database (there is a comparison of non adaptive OT in Figure 5.11).

[9]This is quite easy to achieve by assuming that for all line s, $DB(s) = [db(s)]_1$ where $db(s)$ is the real line (thus known). It implies a few more computation on the user's side in order to recover $db(s)$ from $DB(s)$, but this remains completely feasible if the lines belong to a small space.

Chapter 5

Oblivious Transfer Generalization and New Approach

During the study of Oblivious as a stand alone (e.g. not as an IBE's application) we remarked that an OT and many other protocols such as Oblivious Signature-based Envelope[1] can be generalized into one protocol that we called Oblivious Language-based Envelope. We also give a general instantiation of this new kind of protocol. In this section too SPHF will be a very important tool for our instantiations. Secondly, we link two primitives: Password authenticated key exchange and Oblivious Transfer. Precisely we show how to transform an efficient PAKE into an efficient OT.

5.1 Oblivious Language-based Envelope

In this section we generalize many existing primitives such as Oblivious Signature-Based Envelope, Oblivious Transfer and even Blind IBKEM into a new protocol that we call Oblivious Language-based Envelope. First we describe what is an Oblivious Signature-Based Envelope to see the likelihood with our Oblivious transfer constructed in the previous Chapter. Then we describe our new protocol and give a generic instantiation secure in the UC-framework.

5.1.1 Oblivious Signature-Based Envelope

The construction from 4.3 opens new efficient applications to the already known Oblivious-Transfer protocols. But what happens when someone wants some additional access control by requesting extra properties, like if the user is only allowed to ask two lines with the same parity bits, the user can only request

[1] defined in the next section.

lines for whose number has been signed by an authority, or even finer control provided through credentials? A first example could be with a database made of only one line but the access is delivered only if the recipient owns a special signature. This is called Oblivious Signature-based envelope and has already been studied in [LDB03, BPV12]. In other words in an OSBE protocol a sender $\mathcal{S}$ wants to send a private message $m \in \{0,1\}^{\mathfrak{K}}$ to a recipient $\mathcal{R}$ in possession of a valid certificate/signature on a public message M (given by a certification authority):

Definition 28 (Oblivious Signature-Based Envelope)**.** *An* OSBE *scheme is defined by four algorithms* (Setup, KeyGen, Sign, Verify)*, and one interactive protocol* Protocol$\langle \mathcal{S}, \mathcal{R} \rangle$*:*

- Setup$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*;*
- KeyGen$(\mathfrak{K})$ *generates the keys* $(\mathsf{vk}, \mathsf{sk})$ *of the certification authority;*
- Sign(sk, M) *produces a signature* σ *on the input message* M*, under the signing key* sk*;*
- Verify(vk, M, σ) *checks whether* σ *is a valid signature on* M*, w.r.t. the public key* vk*; it outputs 1 if the signature is valid, and 0 otherwise.*
- Protocol$\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma) \rangle$ *between the sender* $\mathcal{S}$ *with the private message* P*, and the recipient* $\mathcal{R}$ *with a certificate* σ*. If* σ *is a valid signature under* vk *on the common message* M*, then* $\mathcal{R}$ *receives* M*, otherwise it receives nothing. In any case,* $\mathcal{S}$ *does not learn anything.*

An OSBE scheme should fulfill the following security properties [2]. To the best of our knowledge, no UC functionality has already been given.

- *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
- *semantically secure*: the recipient learns nothing about $\mathcal{S}$'s input m if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$ even if he has eavesdropped on several interactions $\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma) \rangle$ with valid signatures, and the same sender's input P;
- *escrow-free* (*oblivious with respect to the authority*): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input.
- *semantically secure w.r.t. the authority*: after the interaction, the authority (owner of the signing key sk) learns nothing about m from a passive access to a challenge transcript[3].

[2] The formal security games are given in [BPV12], in this thesis will formally recall this properties through the UC functionality of OLBE described in figure 5.1.

[3] This property is different from the semantic security described above, indeed in the context of OSBE the signer and the sender can be different entities

5.1.2 Definition of an Oblivious Language-Based Envelope

Oblivious Language-Based Envelope (OLBE) generalizes Oblivious Transfer and OSBE, for n messages (with n polynomial in the security parameter $\mathfrak{K}$) to provide the best of both worlds.

Languages. Let X be a set of elements. The language $\mathfrak{L} \subset X$ used in the definition of an OLBE should be a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element not in $\mathfrak{L}$. We here mainly consider languages which are hard-partitioned subsets, for instance, encryptions of publicly verifiable languages. This sublanguages should fulfill the following properties:

- *Publicly Verifiable:* Given a word x in X along with a witness w, anyone should be able to decide in polynomial time whether $x \in \mathfrak{L}$ or not.
- *Self-Randomizable*: Given a word in the language, anyone should be able to sample a new word in the language[4], and the distribution of this resampling should be indistinguishable from an honestly computed distribution. This will be used in order to prevent an adversary, or the authority in charge of distributing the words, to learn which specific form of the word was used by the user.

In case we consider several languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$, we also assume it is a *Trapdoor Collection of Languages*: It is computationally hard to sample an element in $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_n$, except if one possesses a trapdoor tk (without the knowledge of the potential secret keys)[5]. For example, in the previous chapter we used this property to handle adaptive corruptions.

This concept of languages is very flexible and allows a lot of different applications to the concept of OLBE. For example we can consider a language where it is hard to create a word in the language[6]. This is the case for the OSBE protocol: the signature is given by the authority and under the unforgeability of the signature scheme, it is hard to create a fresh and valid signature.

OLBE. In such a protocol, a sender $\mathcal{S}$ wants to send one or several private messages to a recipient $\mathcal{R}$. $\mathcal{R}$ will be able to recover any message for which he possesses a word in the corresponding language. Let's say that the sender sends $n_{max} \leq n$ for a fixed n. In such a scheme, the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ are assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable.

Following the definitions for OSBE recalled above and given in [LDB03,BPV12], we give the following definition for OLBE. Due to the proximity of the primitives we shape our definition of OLBE as the definition of OSBE.

[4] It should be noted that this property is not incompatible with the potential secret key of the language in case it is keyed (see below).

[5] This implicitly means that the languages are compatible, in the sense that one can indeed find a word belonging to all of them.

[6] We refer to this kind of languages as *Keyed* languages

Definition 29 (Oblivious Language-based Envelope)**.** *An* OLBE *scheme is defined by five algorithms* (Setup, SetupT, KeyGen, Samp, Verify)*, and one interactive protocol* Protocol$\langle\mathcal{S},\mathcal{R}\rangle$*:*

- Setup$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the numbers* n *and* $n_{\max}$*;*

or SetupT$(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[7] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$*.*

- KeyGen(param, $\mathfrak{K}$) *generates, for all* $i\in\{1,\ldots,n\}$*, the description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$*.*
- Samp(param, I) *or* Samp(param, $I,(\mathsf{sk}_{\mathfrak{L}_i})_{i\in I}$) *such that* $I\subset\{1,\ldots,n\}$ *and* $|I|=n_{\max}$*, generates a list of words* $(W_i)_{i\in I}$ *such that* $W_i\in\mathfrak{L}_i$ *for all* $i\in I$*;*
- $\mathsf{Verify}_i(W_i,\mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*
- Protocol$\langle\mathcal{S}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(m_1,\ldots,m_n)),\mathcal{R}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(W_i)_{i\in I})\rangle$*, which is executed between the sender* $\mathcal{S}$ *with the private messages* $(m_1,\ldots,m_n)$ *and corresponding languages* $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$*, and the recipient* $\mathcal{R}$ *with the same languages and the words* $(W_i)_{i\in I}$ *with* $I\subset\{1,\ldots,n\}$ *and* $|I|=n_{\max}$*, proceeds as follows. For all* $i\in I$*, if the algorithm* $\mathsf{Verify}_i(W_i,\mathfrak{L}_i)$ *returns* 1*, then* $\mathcal{R}$ *receives* m_i*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

The collections of words can be a single certificate/signature on a message M (encompassing OSBE, with $n=n_{\max}=1$), a password, a credential, an equivocable commitment of a line number (encompassing our UC secure 1-out-of-n oblivious transfer, with $n_{\max}=1$), k line numbers (encompassing k-out-of-n oblivious transfer, with $n_{\max}=k$), etc... Like OSBE and our adaptive OT, we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors.

5.1.3 Security Properties and Ideal Functionality of OLBE

We describe here the properties that an OLBE should satisfy, we further describe them more formally with an ideal functionality (see Figure 5.1):

- *correctness*: the protocol actually allows $\mathcal{R}$ to learn $(m_i)_{i\in I}$, whenever $(W_i)_{i\in I}$ are valid words of the languages $(\mathfrak{L}_i)_{i\in I}$, where $I\subset\{1,\ldots,n\}$ and $|I|=n_{\max}$;
- *semantically secure* (*sem*): the recipient learns nothing about the input m_i of $\mathcal{S}$ if it does not use a word in $\mathfrak{L}_i$. More precisely, if $\mathcal{S}_0$ owns $m_{i,0}$ and $\mathcal{S}_1$ owns $m_{i,1}$, the recipient that does not use a word in $\mathfrak{L}_i$ cannot

[7] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

distinguish between an interaction with $\mathcal{S}_0$ and an interaction with $\mathcal{S}_1$ even if the receiver has seen several interactions

$$\langle \mathcal{S}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (m_1, \ldots, m_n)), \mathcal{R}((\mathfrak{L}_1, \ldots, \mathfrak{L}_n), (W'_j)_{j \in I}) \rangle$$

with valid words $W'_i \in \mathfrak{L}_i$, and the same sender's input m_i;

- *escrow free* (*oblivious with respect to the authority*): the authority corresponding to the language $\mathfrak{L}_i$ (owner of the language secret key $\mathsf{sk}_{\mathfrak{L}_i}$ – if it exists), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a word W_i in the language $\mathfrak{L}_i$ or not. This requirement also holds for anyone holding the trapdoor key tk.
- *semantically secure w.r.t. the authority* (sem^*): after the interaction, the trusted authority (owner of the language secret keys if they exist) learns nothing about the values $(m_i)_{i \in I}$ from the transcript of the execution. This requirement also holds for anyone holding the trapdoor key tk.

Moreover, the setup algorithms should be indistinguishable and it should be infeasible to find a word belonging to two or more languages without the knowledge of tk.

The functionality $\mathcal{F}_{\text{OLBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1, \ldots, \mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving from party $\mathfrak{P}_i$ an input of the form $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$**, with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further `Send`-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input of the form $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (W_i)_{i \in I})$ where $I \subset \{1, \ldots, n\}$ and $|I| = n_{\max}$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m'_k)_{k \in I})$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verify}_k(W_k, \mathfrak{L}_k)$ returns 1, and $m'_k = \perp$ otherwise. Ignore further `Received`-message with the same ssid from $\mathfrak{P}_j$.

Figure 5.1: Ideal Functionality for Oblivious Language-Based Envelope $\mathcal{F}_{\text{OLBE}}$

The ideal functionality is inspired from the oblivious transfer functionality given in [Can01, CKWZ13, ABB+13]. As for oblivious transfer (Figure 2.16), we adapt them to the simple UC framework for simplicity.

Remark 6. *If we specify the language to a set of signature over a known message this ideal functionality becomes an ideal functionality for an* OSBE *scheme which has never been defined before.*

5.1.4 Generic UC-Secure Instantiation of OLBE with Adaptive Security

For simplicity we focus on the case $n_{\max} = 1$. It encompasses OSBE and 1-out-of-n OT. To get the more general cases one could run many $n_{\max} = 1$-OLBE in parallel. This simplify the algorithms Samp and Verify.

Building Blocks. We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\text{cca}}, \mathsf{KeyGen}_{\text{cca}}, \mathsf{Encrypt}^{\ell}_{\text{cca}}, \mathsf{Decrypt}^{\ell}_{\text{cca}})$[8] compatible with an SPHF onto a set G. In the KeyGen algorithm, the description of the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ thus implicitly defines the languages $(\mathfrak{L}^c_1, \ldots, \mathfrak{L}^c_n)$ of CCA-encryptions of elements of the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$.

We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in G$.

We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\text{cpa}}, \mathsf{KeyGen}_{\text{cpa}}, \mathsf{Encrypt}_{\text{cpa}}, \mathsf{Decrypt}_{\text{cpa}})$ with plaintext size at least equal to the security parameter.

We follow the ideas of the oblivious transfer constructions given in [ABB+13, BC15], giving the protocol presented on Figure 5.2. For the sake of simplicity, we only give the version for adaptive security, in which the sender generates a public key pk and ciphertext c to create a somewhat secure channel (they would not be used in the static version).

All the instantiations given are pairing-based but techniques explained in [BC15] could be used to rely on other families of assumptions, like decisional quadratic residue or even LWE.

Theorem 11. *The oblivious language-based envelope scheme described in Figure 5.2 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures, an* IND-CPA *encryption scheme, and an* IND-CCA *encryption scheme admitting an* SPHF *on the language of valid ciphertexts of elements of* $\mathfrak{L}_i$ *for all* i[9].

The proof of this theorem is close to the proof of the Theorem 10, the main difference comes from the fact that we do not have the blind IBE construction. The other difference is the nature of the protocols: the language for Oblivious transfer is precise (commitments of a number of line), but in the OLBE it is general. The techniques remain the same for the rest of the proof.

[8] the notion of labeled encryption is very close to a simple encryption. In a labelled encryption the algorithm Encrypt and Decrypt take as input a label. A ciphertext encrypted with a label ℓ can only be decrypted with the decryption algorithm using the same label.

[9] as soon as the languages are self-randomizable, publicly-verifiable and admit a common trapdoor

CRS: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cca}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cca}}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Pre-flow:

1. Sender generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
2. Sender sends pk to User.

Flow From the Receiver $\mathcal{R}$:

1. User chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. User computes $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$.
3. User completely erases J and the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Sender. He also checks the validity of his words: the receiver only keeps the random coins used by $\mathsf{Encrypt}_{\mathrm{cca}}$ for the j such that $\mathsf{Verify}_j(W, \mathfrak{L}_j) = 1$ (since he knows they will be useless otherwise).

Flow From the Sender $\mathcal{S}$:

1. Sender decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $R \leftarrow F(J)$.
2. For all $j \in \{1, \ldots, n\}$, sender computes $\mathsf{hk}_j = \mathsf{HashKG}(\ell, \mathfrak{L}^c_j, \mathsf{param})$, $\mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, \ell, (\mathfrak{L}^c_j, \mathsf{param}))$, $v_j = \mathsf{Hash}(\mathsf{hk}_j, (\mathfrak{L}^c_j, \mathsf{param}), (\ell, C))$, $Q_j = m_j \oplus \mathtt{KDF}(v_j) \oplus R$.
3. Sender erases everything except $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$ and sends them over a secure channel.

Message recovery:
Upon receiving $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$, $\mathcal{R}$ can recover m_i by computing $m_i = Q_i \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}^c_i, \mathsf{param}), (\ell, C), r) \oplus R$.

Figure 5.2: UC-Secure OLBE for One Message (Secure Against Adaptive Corruptions)

Proof. We prove the adaptive security of this protocol via a sequence of games, starting from the real game, where the adversary $\mathscr{A}$ interacts with the real players, and ending with the ideal game, where we have built a simulator $\mathscr{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathscr{A}$.

The main idea is that, by assumption, the simulator can always obtain the common trapdoor tk of the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ and use it to commit to a word simultaneously belonging to all the languages. In case of adaptive corruption, we face two cases: either the word was not correct, in which case, following the real protocol, the user should have erased his randomness, so that the simulator does not have anything to reveal. Or the word was correct and should belong to a certain language $\mathfrak{L}_i$, and the simulator can then adapt the word and randomness so that it seems to belong to $\mathfrak{L}_i$ (only). This enables us to avoid the use of commitments both extractable and equivocable (which is the usual tool for adaptive corruptions).

We say that a flow is *oracle-generated* if it was sent by an honest player (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game G_1: This is the real game.

Game G_2: In this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the inputs $(m_1, \ldots, m_n)$ and W sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game G_3: In this game, we replace the setup algorithm Setup by SetupT, allowing the existence of a trapdoor to find words in the intersection of the collection of languages. We also allow the simulator to program $\mathsf{Setup}_{\mathsf{CCA}}$ in the CRS, enabling it to learn the extraction trapdoor of the CCA encryption scheme. The indistinguishability of the setup makes this game indistinguishable from the former one for the environment. Corruptions are handled the same way.

Game G_4: We first deal with oracle-generated flows from the senders $\mathcal{S}$: when receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers i such that $W \in \mathfrak{L}_i$. If it recovers $i \neq j$ such that $W \in \mathfrak{L}_i \cap \mathfrak{L}_j$, then the adversary has broken the infeasibility of finding a word in an intersection of languages without knowing the trapdoor and we abort the game. Otherwise, instead of computing the key v_t, for $t = 1, \ldots, n$ with the hash function, the simulator then chooses $v_t \xleftarrow{\$} G$ for $t \neq i$.

With an hybrid proof, applying the smoothness for every honest sender, on every index $t \neq i$, since C is extracted to $W \in \mathfrak{L}_i$, for any $t \neq i$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly). This game is thus indistinguishable from the previous one under the smoothness.

Game G_5: Still in this case, when receiving a commitment C, the simulator extracts the committed value W, giving it the number i. Instead of proceeding as the sender would do on $(m_1, \ldots, m_n)$, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_i = m_i$, but $m'_t = 0$ for all $t \neq i$. Since the masks v_t, for $t \neq i$, are random, this game is perfectly indistinguishable from the previous one.

Game G_6: We now deal with oracle-generated flows from the receivers $\mathcal{R}$: the simulator now uses the trapdoor tk to find a word W in the intersection of all languages.

With an hybrid proof, applying a security game in each session in which the simulator does not know the trapdoor tk, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value i.

Game G_7: We deal with **the generation of R for honest senders $\mathcal{S}$ on oracle-generated queries (adaptive case only)**: if $\mathcal{S}$ and $\mathcal{R}$ are honest at least until $\mathcal{S}$ received the second flow, the simulator sets $R = F(J')$ for both $\mathcal{S}$ and $\mathcal{R}$, with J' a random value, instead of $R = F(J)$.

With an hybrid proof, applying the $\mathsf{IND\text{-}CPA}$ property for each session, one can show the indistinguishability of this game with the previous one.

Game G_8: Still in the same case, the simulator sets R as a random value, instead of $R = F(J')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game G_9: We now deal with **the generation of v_i for honest senders $\mathcal{S}$ on oracle-generated queries**:

- in the static case (the pre-flow is only needed to compute $(\mathsf{vk}, \mathsf{vtk})$, and thus we assume $R = 0$) the simulator chooses $v_i \xleftarrow{\$} G$ (for $t \neq i$, the simulator already chooses $v_t \xleftarrow{\$} G$), where i is the index given by the ideal functionality to the honest receiver $\mathcal{R}$.

  With an hybrid proof, applying the pseudo-randomness for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C;

- in the adaptive case, and thus with the additional random mask R, one can send a random m_i, and v_i can be computed later (when $\mathcal{R}$ actually receives its flow).

  As above, but only if $\mathcal{R}$ has not been corrupted before receiving its flow, the simulator chooses $v_s \xleftarrow{\$} G$. With an hybrid proof, applying the

pseudo-randomness, for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C. If the player $\mathcal{R}$ involved in the pseudo-randomness game gets corrupted (but the decommitment information is unknown) we are not in this case, and we can thus abort it.

In case of corruption of $\mathcal{S}$, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly).

In case of corruption of the receiver $\mathcal{R}$, and thus receiving the value m_i, the simulator chooses R (because it was a random value unknown to the adversary and all the other v_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (L_i, \mathsf{param}), (\ell, C), r_i) \oplus m_i = Q_i.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game G_{10}: Still in this case, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_t = 0$ for all i. Since the masks $v_t \oplus R$, for any $t = 1, \ldots, n$, are independent random values (the v_t, for $t \neq i$ are independent random values, and v_i is also independently random in the static case, while R is independently random in the adaptive case), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index i given by the ideal functionality to the honest receiver $\mathcal{R}$, to simulate $\mathcal{S}$ (but it is still necessary to simulate $\mathcal{R}$).

Game G_{11}: We do not use anymore the knowledge of i when simulating an **honest receiver** $\mathcal{R}$: the simulator generates a word W in the intersection of the languages and $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; r)$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$, to send C during the first phase of honest receivers. This does not change anything from the previous game since the randomness needed to open to a word in another language is never revealed.

When it thereafter receives $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R}, (Q_1, \mathsf{hp}_1, \ldots, Q_n, \mathsf{hp}_n))$ from the adversary, the simulator computes, for $i = 1, \ldots, k$, r_i,

$$v_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i)$$

and $m_i = v_i \oplus R \oplus Q_i$. This provides the database submitted by the sender.

Game G_{12}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionality, which means that an honest sender has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;

- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent a first flow, the simulator generates a word W in the intersection of languages, $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$ and $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, J)$ where R is a random value;
- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value W and recovers i (aborting in case of multiple values), and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as J, and computes $R = F(J)$);
- when receiving $(Q_1, \mathsf{hp}_1, \ldots, Q_n, \mathsf{hp}_n)$ from the adversary (a corrupted sender), the simulator computes, for $i = 1, \ldots, n$, r_i,

  $$v_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i)$$

  and $m_i = v_i \oplus R \oplus Q_i$. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with m_i, on behalf of a corrupted receiver, from the extracted W leading to i, instead of proceeding as the sender would do on $(m_1, \ldots, m_n)$, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_i = m_i$, but $m'_j = 0$ for all $j \neq i$;
- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(m'_1, \ldots, m'_n)$, with $m'_j = 0$ for all j, but it chooses R uniformly at random instead of choosing it as $R = F(J)$; in case of corruption afterward, the simulator will adapt R such that $R \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i) \oplus Q_i = m_i$, where m_i is the message actually received by the receiver.

Any corruption either reveals i earlier, which allows a correct simulation of the receiver, or reveals $(m_1, \ldots, m_n)$ earlier, which allows a correct simulation of the sender. When the sender has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the receiver is corrupted after the sender has sent is flow, but before the receiver receives it, since he has kept r_i: this would enable the adversary to recover m_i from Q_i and hp_i. This is the goal of the ephemeral mask R that provides a secure channel.

□

5.1.5 Oblivious Primitives Obtained by the Framework

More than the two examples already discussed in this chapter, the framework also enables us to give a new instantiation of Access Controlled Oblivious Transfer under classical assumptions [CDN09]. In such a primitive, the user does not

automatically gets the line he asks for, but has to prove that he possesses one of the credential needed to access this particular line.

5.2 Very Efficient Oblivious Transfer

The key block needed for our construction of an oblivious transfer/OLBE is the Smooth Projective Hash Functions. It allows to compute a value by two different ways. We can notice that the 2 parties never know if they computed the same value or not. SPHF are heavy tools, very powerful but they come with high computation and communication costs. In this section we investigate a way to replace the role of the SPHF with a lighter primitive. We will first construct OT from PAKE. Indeed PAKE fullfil the property discussed before (the values are equal if they used the same password) and there exists instantiations of PAKE that does not involve any SPHF [JR15].

A second motivation is the willingness to link Oblivious Transfer and PAKE. Finding the link between primitives and showing which primitives can be used to build other ones is a useful trend in proven cryptography. This has been done many times before (IBE-Signature,IBE-IBS,...) and even during this PhD (ABE-IBE,IBE-Fragmented Blind IBE, Fragmented Blind IBE-OT). We remark that many transformations involved IBE confirming the central place of this primitive.

This section will be divided into 4 parts. First we will define the primitive PAKE and the security requirements for it. Then we will see how to generically transform a PAKE into an Oblivious Transfer and afterwards apply it on different examples. Finally we will apply our transformation on a QA-NIZK-based PAKE to obtain our very efficient OT.

5.2.1 Password Authenticated Key Exchange

Roughly speaking a PAKE is an interactive protocol allowing two parties to share a common secret by using passwords.

We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 5.3. It was described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of a wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

The functionality $\mathcal{F}_{\text{pwKE}}$ is parametrized 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, ssid, P_i, P_j, pw) from party P_i:**

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

- **Upon receiving a query (`TestPwd`, sid, ssid, P_i, pw′) from the adversary $\mathcal{S}$:**

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

- **Upon receiving a query (`NewKey`, sid, ssid, P_i, sk) from the adversary $\mathcal{S}$:**

  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$, 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 $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw') with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ as `completed`.

Figure 5.3: Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

The passwords are supposed to be distributed before the protocol, our security holds for every distribution of passwords.

Before the UC framework, the classical security model for PAKE protocols was the BPR model [BPR00]. The first ideal functionality for PAKE protocols in the UC framework [Can01, CK02] was proposed by Canetti *et al.* [CHK+05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol[10].

In case of corruption, the adversary learns the password of the corrupted player; After the `NewKey`-query, it additionally learns the session key.

Description of a UC-Secure PAKE Scheme

For sake of simplicity we are going to consider PAKE protocols in two flows, since there already exists a multitude of PAKE schemes satisfying this requirement, and since minimizing the number of flows is one of the most important issues in modern instantiations. This allows us to generically give an optimization, leading to more efficient protocols.

In order to present the framework of our transformation, we formally split a PAKE scheme into the following four algorithms (note that many algorithm use two random element, one for each password): .

- Setup: An algorithm that generates the initial crs.
- $\mathsf{flow}_1(\mathsf{pw}_i; \rho_i, \rho'_i)$: The algorithm run by the user P_i, using some randomness ρ_i and ρ'_i and possessing the password pw_i, to generate the first flow of the protocol. We note ρ_i the part of the randomness involved in hiding the password, and ρ'_i the remainder.
- $\mathsf{flow}_2(\mathsf{flow}_1, \mathsf{pw}_j; \rho_j, \rho'_j)$: The algorithm run by the user P_j, using some randomness ρ_j and possessing the password pw_j, after receiving the flow flow_1, to generate the second flow of the protocol. This algorithm also produces P_j's view of the key: $K_i^{(j)}$.
- $\mathsf{Decap}(\mathsf{flow}_2, \mathsf{pw}_i, f(\rho_i, \rho'_i))$: The algorithm run by P_i when receiving the flow flow_2 from P_j, using his password pw_i and a function $f(\rho_i, \rho'_i)$ of the randomness initially used and the remainder (which function can be anything from the identity to the null function), to recover P_i's view of the key: $K_j^{(i)}$.

5.2.2 Generic Construction of a UC-Secure OT Scheme

In Canetti et al. in [CDVW12], they show how a UC-secure Oblivious Transfer scheme can be transformed into a UC-secure PAKE scheme. In this section we are going to go the other way, and show how a UC-secure PAKE can be transformed into a UC-secure Oblivious Transfer scheme. While considering this direction may seem like transforming a very strong primitive into a weaker one, this will

[10] we will later apply our transformation to a PAKE secure in the BPR model to show that our transformation can apply also in a lattice-based instantiation

allow us to prepare a framework to propose the most efficient Oblivious Transfer scheme to date.

We denote by $\mathsf{DB} = (L_1, \ldots, L_n)$ the database of the server containing n lines, and by s the line requested by the user in an oblivious way. As for OT and OLBE, we assume the existence of a CPA-Secure encryption scheme ($\mathsf{Setup}_{\text{cpa}}$, , $\mathsf{KeyGen}_{\text{cpa}}, \mathsf{Encrypt}_{\text{cpa}}, \mathsf{Decrypt}_{\text{cpa}}$) [11] and the existence of a Pseudo-Random Generator (PRG) F. The public parameters of the encryption scheme are going to be included in the public common reference string crs generated by the setup algorithm of the PAKE scheme.

Using the PAKE scheme, the encryption scheme and the PRG, one can now obtain an OT scheme from a PAKE scheme, as described in Figure 5.4. The idea of the protocol is as follows:

- First, a setup phase enables to generate the CRS, both for the PAKE and encryption schemes.
- The pre-flow is a technical requirement for the UC proof.
- When querying for line s, the receiver proceeds in three steps. The first one consists in generating a masking value R (technical requirement for the UC proof), the second one consists in running the first flow of the PAKE scheme for password s to generate the first flow denoted [12] as flow_0, and the third one consists in erasing anything but the needed values, and sending flow_0 to the sender.
- When answering to this query, the sender also proceeds in three steps. The first one consists in recovering the value R and the second one consists in running, for each line $k \in \{1, \ldots, n\}$, the second flow of the PAKE scheme to generate the second flow denoted as flow_k (using k as the password) as well as $\mathcal{S}$'s view of the session key, denoted as $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$. Finally, in the third step, the server computes a xor Q_k of the line L_k and this session key and the value R and sends the set (flow_k, Q_k) back to $\mathcal{R}$.
- When receiving this set of values, the receiver uses flow_s to run the decapsulation algorithm of the PAKE scheme (with password s) to recover his view of the session key, denoted as $(K_s)_{\mathcal{S}}^{(\mathcal{R})}$. He finally uses R, Q_s and $(K_s)_{\mathcal{S}}^{(\mathcal{R})}$ to recover the expected line L_s.

The correctness easily follows from the correctness of the PAKE protocol, since the views of $\mathcal{R}$ and $\mathcal{S}$ of the session key for the PAKE scheme executed for password s are the same. The scheme is oblivious with respect to $\mathcal{S}$ since the first flow of the PAKE scheme executed by $\mathcal{R}$ does not reveal any information on s. And it is oblivious with respect to $\mathcal{R}$ since the correctness of the PAKE scheme gives him no information on the session keys $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$ for $k \neq s$. The security is stated in Theorem 12.

[11] Here again this encryption scheme handle corruptions before the first flow.

[12] Note that we now denote as flow_0 (and not flow_1) the flow generated by the PAKE algorithm flow_1, in order to avoid the confusion with the flow_1 message generated by the sender (for line 1) during the database answer phase.

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^{\hat{\kappa}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\hat{\kappa}})$.

Pre-Flow:

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:

1. $\mathcal{R}$ picks a random J, computes $R \leftarrow F(J)$ and sets $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. $\mathcal{R}$ picks a random (ρ, ρ'), and runs $\mathsf{flow}_1(s; \rho, \rho')$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \rho')$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(J)$.
2. For each line k:
   - $\mathcal{S}$ picks a random (ρ_k, ρ'_k), runs $\mathsf{flow}_2(\mathsf{flow}_0, k; \rho_k, \rho'_k)$ to generate flow_k and $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$.
   - $\mathcal{S}$ then computes $Q_k = L_k \oplus (K_k)_{\mathcal{R}}^{(\mathcal{S})} \oplus R$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_k, Q_k)_{k \in [\![1,n]\!]}$ and sends it to $\mathcal{R}$.

Data recovery:
$\mathcal{R}$ then using $f(\rho, \perp)$ computes $(K_s)_{\mathcal{S}}^{(\mathcal{R})} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \rho'))$, sets $L_s = Q_s \oplus (K_s)_{\mathcal{S}}^{(\mathcal{R})} \oplus R$, and erases everything else.

Figure 5.4: UC-Secure 1-out-of-n OT from a UC-Secure PAKE

Theorem 12. *Under the* UC*-security*[13] *of the* PAKE *protocol, the existence of a Pseudo-Random Generator (*PRG*) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database, and the* IND-CPA *security of the encryption scheme, the transformation presented in Figure 5.4 achieves a* UC*-secure 1-out-of-n Oblivious Transfer scheme, with the same handling of corruptions as in the* PAKE *protocol.*

The high-level idea of the proof is quite simple. Recall that in a UC proof, one has to simulate one of the two players in front of a corrupted player. The first flow allows the simulated server to extract the requested line. One will then be able to send random noise on all the lines besides the correct one. The PAKE functionality ensures that this is possible. Then again for honest users, the simulator will replace the remaining line by a random noise. In case of corruptions of the server, one relies on the security of the CPA encryption, to ensure that everything was done honestly (the adversary is not able to test the validity of the retrieved R, and so for the correct line as $L \oplus K \oplus R$ where L and K are the honest values, and R is simply set as $Q \ominus (L \oplus K)$). In case of corruption of the user, we rely on the PAKE handling of user corruption to adapt the memory.

Proof. To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathscr{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathscr{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathscr{A}$. We prove the adaptive version of the protocol. The proof of the static framework version can be obtained by removing the parts related to adaptive version from the proof below.

When the sender submits its values, the simulator can extract all the message thanks to the crs trapdoor and get the witnesses for each indices. This allows to simulate the `Send`-query to the ideal functionality.

Eventually, when simulating the honest senders, the simulator recovers the expected line value s from flow_0, to set flow_s and L_s honestly, the other values can be random. More details follow:

Game G_1: This is the real game.

Game G_2: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs $(L_1, \ldots, L_n)$ and s sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

[13] Note that the transformation also allows to obtain an OT scheme from a BPR-secure PAKE scheme in a game-based security model, but since this is of less interest, we do not formally prove it due to lack of space.

Game G_3: In this game, we just replace the Setup algorithm (if any), by the one used in the PAKE proof, as this is transparent in the PAKE protocol, this is also transparent in the Oblivious Transfer.

Game G_4: We first deal with **honest senders** P_i: when receiving a first flow flow_0, the simulator recovers the prospective line value s (this is because, we assume the UC-PAKE secure, hence after flow_1, the simulator should be able to send $\mathtt{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, s$, hence knows said s). Instead of computing the key K_t, for $t = 1, \ldots, n$ for all possible line/passwords, it chooses K_t at random in the key space for $t \neq s$.

With an hybrid proof, for every honest sender, on every index $t \neq s$, since flow_0 leads to s, for any $t \neq s$, the new key is indistinguishable from a random value, under the PAKE security.

In case of corruption, everything is either already handled by the PAKE, or erased. This game is thus indistinguishable from the previous one under the PAKE functionality.

Game G_5: Still in this case, when receiving flow_0, the simulator recovers the expected value s. Instead of proceeding as the sender would do on lines $(L_1, \ldots, L_n)$, the simulator proceeds on $(L'_1, \ldots, L'_n)$, with $L'_s = L_s$, but $m'_t = 0$ for all $t \neq s$. Since the masks K_t, for $t \neq s$, are random, this game is perfectly indistinguishable from the previous one.

Game G_6: We now deal with **honest receivers** P_j: we replace flow_1 for in Step 1 of the index query phase of honest receivers by simulated flow_1, using the fact that the UC-Proof requires the simulator at this step to send a non committing flow compatible with every possible password, and keep the PAKE memory.

Using the PAKE security, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value s.

Game G_7: We deal with **the generation of R for honest senders P_i on honestly-generated queries (adaptive case only)**: if P_i and P_j are honest at least until P_i received the second flow, the simulator sets $R = F(S')$ for both P_i and P_j, with S' a random value, instead of $R = F(S)$.

With an hybrid proof, applying the IND-CPA property for each session, one can show the indistinguishability of this game with the previous one.

Game G_8: Still in the same case, the simulator sets R as a random value, instead of $R = F(S')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game G_9: We now deal with **the generation of K_s for honest senders P_i on honestly-generated queries**:

- in the static case (the pre-flow is not necessary, and thus we assume $R = 0$) the simulator chooses K_s at random in the Key Space (for $t \neq s$, the simulator already chooses K_t random), where s is the index given by the ideal functionality to the honest receiver P_j.
  Under the PAKE security, for every honest sender, the key value is indistinguishable from a random value, because the adversary does not know any secret information on flow_0;
- in the adaptive case, and thus with the additional random mask R, one can send a random Q_s, and K_s can be computed later (when P_j actually receives its flow, and the simulator possibly learns the expected L_s).
  As above, but only if P_j has not been corrupted before receiving its flow, the simulator chooses K_s at random. Once again, invoking the PAKE security, and the lack of secret information known by the adversary, this value is indistinguishable from a random value. If the player P_j gets corrupted we are not in this case, and we can thus abort it.
  In case of corruption of P_i, everything has been erased and / or handled by the PAKE simulator. In case of corruption of the receiver P_j, and thus receiving the value L_s, the simulator chooses R (because it was a random value unknown to the adversary and all the other K_t are independent random values too) such that $R \oplus K_s \oplus Q_s = L_s$.

This game is thus indistinguishable from the previous one under the PAKE Security (External Adversary, Adaptive Corruption of the second User).

Game G_{10}: Still in this case, the simulator proceeds on $(L'_1, \ldots, L'_n)$, with $L'_t = 0$ for all t. Since the masks $K_t \oplus R$, for any $t = 1, \ldots, n$, are independent random values (the K_t, for $t \neq s$ are independent random values, and K_s is also independently random in the static case, while R is independently random in the adaptive case), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index s given by the ideal functionality to the honest receiver P_j, to simulate P_i (but it is still necessary to simulate P_j).

Game G_{11}: We do not use anymore the knowledge of s when simulating an **honest receiver** P_j: the simulator generates an equivocable first flow flow_0 during the index query phase of honest receivers. We essentially break the atomic flow_1 from the PAKE in the two separated processes SimFlow and OpenFlow, used by the simulator in the PAKE proof, and stores the equivocation information Λ, instead of just $f(\rho)$. This does not change anything from the previous game since the memory before flow_0 is never revealed. It can be updated with correct values in case of corruption of the receiver.

When it thereafter receives the message $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (Q_1, \ldots, Q_n))$ from the adversary, the simulator computes, for $i = 1, \ldots, n$, the value $f(\rho_i) \leftarrow \mathsf{OpenFlow}^{\ell}(\mathsf{flow}_0, i, \Lambda)$, $K_i \leftarrow \mathsf{Decap}(\mathsf{flow}_i, i, f(\rho_i))$ and $m_i = K_i \oplus R \oplus Q_i$. This provides the database submitted by the sender.

Game G_{12}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionality, which means that an honest sender has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent an index query, the simulator generates $(\mathsf{flow}_0, \Lambda) \xleftarrow{\$} \mathsf{SimFlow}^{\ell}()$ and $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ and R a random value, to send flow_1 and c during the index query phase of the honest receiver;
- when receiving flow_0 and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator recovers the committed value s, and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as S, and computes $R = F(S)$);
- when receiving $(\mathcal{Q}_1, \ldots, Q_n)$ from the adversary (a corrupted sender), the simulator computes, for $i = 1, \ldots, n$, $f(\rho_i) \leftarrow \mathsf{OpenFlow}^{\ell}(\mathsf{flow}_0, i, \Lambda)$, $K_i \leftarrow \mathsf{Decap}(\mathsf{flow}_i, i, f(\rho_i))$ and $L_i = K_i \oplus R \oplus Q_i$. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with L_s, on behalf of a corrupted receiver, from the recovered s, instead of proceeding as the sender would do on $(L_1, \ldots, L_n)$, the simulator proceeds on $(L'_1, \ldots, L'_n)$, with $L'_s = L_s$, but $L'_i = 0$ for all $i \neq s$;
- when receiving a a flow flow_0 and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(L'_1, \ldots, L'_k)$, with $L'_i = 0$ for all i, but it chooses R uniformly at random instead of choosing it as $R = F(S)$; in case of corruption afterward, the simulator will adapt R such that $R \oplus K_s \oplus Q_s = L_s$, where L_s is the message actually received by the receiver.

Any corruption either reveals s earlier, which allows a correct simulation of the receiver, or reveals $(L_1, \ldots, L_n)$ earlier, which allows a correct simulation of the sender. When the sender has sent his flow, he has already erased all his random coins (or at least enough to be compatible with the PAKE key). However, there would have been an issue when the receiver is corrupted after the sender has sent is flow, but before the receiver receives it, since he has kept some witness $f(\rho)$: this would enable the adversary to recover L_s from Q_s. This is the goal of the ephemeral mask R that provides a secure channel.

As a consequence, the distance between the first and the last games is bounded by q times the PAKE security, and the encryption CPA indistinguishability.

□

Generic Optimization

In the previous transformation we never used the fact that the first user in the PAKE scheme (here, the receiver) does not learn the password from the second

(here, the sender). This is a property required by PAKE but not useful in the transformation (since the sender executes n parallel PAKE schemes, using the (public) number k of the lines as the password). This argument is the key of the optimization we now propose in Figure 5.5. The difference is that in the index query, the receiver does not pick the second randomness ρ' and in the database answer, the randomness ρ_k is not used anymore. The proof remains the same as the previous one.

5.2.3 A Warm Up

Since in [ABB+13], the authors proposed both a UC-secure PAKE and a UC-secure Oblivious Transfer constructions, this section serves as an example of application of our framework. More precisely, we apply here our (optimized) transformation to their UC-secure PAKE and show that we obtain exactly their UC-secure Oblivious Transfer (which is the Oblivious Transfer that inspired the instantiation of OLBE in the previous Chapter).

Description of Their UC-Secure PAKE Scheme

Description of the scheme (see Figure 5.6): Both parties (denoted as P_b and $P_{\overline{b}}$) want to share a key by using a common password pw. This will thus fix a language $\mathfrak{L}_{\mathsf{pw}}$, which is the set of valid commitments (the linear encryption in their instantiation) of pw, for a smooth projective hash function. Both flows are done simultaneously. In flow_b, each party P_b will generate a pair of hash keys $(\mathsf{hk}_b, \mathsf{hp}_b)$ and then commit to the password pw_b in C_b. Note that P_b stores the witness δ_b that this value C_b is a valid commitment of pw_b and sends (hp_b, C_b) to the other party.

In the decapsulation phase, each party P_b then receives the commitment and the projection key of the other party and computes two hash values: On the one hand, the projected hash value using his own commitment C_b along with the witness δ_b and the projected key $\mathsf{hp}_{\overline{b}}$ sent by the other party, and on the other hand, the (regular) hash value using the received commitment $C_{\overline{b}}$ and the secret hash key hk_b. The shared session key is then defined as the product of these two values.

The commitment scheme used has specific properties explained in [ABB+13] but we interest only needs it to be compatible with SPHF, and both extractable and equivocable.

Completeness: If the two parties used the same password pw, then at the end of the protocol they will share the same value. Thanks to the property of the SPHF, the hash value and the projected value (on the same commitment C_b) will be equal because the commitment belongs to the good language $\mathfrak{L}_{\mathsf{pw}}$ with the good witness δ_b.

Applying the Framework to Obtain a UC-Secure OT Scheme

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^{\lambda}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\lambda})$.

Pre-Flow:

1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:

1. $\mathcal{R}$ picks a random J, computes $R \leftarrow F(J)$ and sets $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. $\mathcal{R}$ picks a random ρ, and runs $\mathsf{flow}_1(s; \rho, \perp)$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \perp)$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:

1. $\mathcal{S}$ decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(J)$.
2. For each line k:
   - $\mathcal{S}$ picks a random ρ'_k, and runs $\mathsf{flow}_2(\mathsf{flow}_0, k; \perp, \rho'_k)$ to generate flow_k and $(K_k)_{\mathcal{R}}^{(\mathcal{S})}$.
   - $\mathcal{S}$ then computes $Q_k = L_k \oplus (K_k)_{\mathcal{R}}^{(\mathcal{S})} \oplus R$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_k, Q_k)_{k \in [\![1,n]\!]}$ and sends it to $\mathcal{R}$.

Data recovery: $\mathcal{R}$ then using $f(\rho, \perp)$ computes $(K_s)_{\mathcal{S}}^{(\mathcal{R})} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \perp))$, sets $L_s = Q_s \oplus (K_s)_{\mathcal{S}}^{(\mathcal{R})} \oplus R$, and erases everything else.

Figure 5.5: Optimized UC-Secure 1-out-of-n OT from a UC-Secure PAKE

- Setup: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$[a].
- $\mathsf{flow}_b(\mathsf{pw}_b; \rho_b)$:
  - Generates $\mathsf{hk}_b \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_{\mathsf{pw}_b})$, $\mathsf{hp}_b \leftarrow \mathsf{ProjKG}(\mathsf{hk}_b, \mathfrak{L}_{\mathsf{pw}_b}, \perp)$
  - picks additional ρ'_b and computes $(C_b, \delta_b) \xleftarrow{\$} \mathsf{Com}^{\ell_b}(\mathsf{pw}_b; \rho'_b)$ with $\ell_b = (\mathsf{sid}, \mathsf{ssid}, P_b, P_{\bar{b}}, \mathsf{hp}_b)$. Sets $\rho_b = (\mathsf{hk}_b, \rho'_b)$.
  - stores hk_b, δ_b, completely erases random coins used before and sends (hp_b, C_b).
- $\mathsf{Decap}(\mathsf{flow}_b, \mathsf{pw}_b, f(\rho_b) = \mathsf{hk}_b, \delta_b)$:
  - Computes $H'_b \leftarrow \mathsf{ProjHash}(\mathsf{hp}_{\bar{b}}, \mathfrak{L}_{\mathsf{pw}_b}, (\ell_b, C_b), \delta_b)$ and $H_{\bar{b}} \leftarrow \mathsf{Hash}(\mathsf{hk}_b, \mathfrak{L}_{\mathsf{pw}_b}, (\ell_{\bar{b}}, C_{\bar{b}}))$ with $\ell_{\bar{b}} = (\mathsf{sid}, \mathsf{ssid}, P_{\bar{b}}, P_b, \mathsf{hp}_{\bar{b}})$
  - Computes $\mathsf{sk}_b = H'_b \cdot H_{\bar{b}}$ and erases everything else, except pw_b.

[a] It makes use of linear encryption as commitment and and use their SPHF based on the language: linear encryption of valid password.

Figure 5.6: UC-Secure PAKE from [ABB+13]

We described in section 2.8.2 the Oblivious Transfer from [ABB+13]. In this paper they proposed a PAKE and an OT as applications to their SPHF. With our (optimized) transformation we manage to come from one to the other.

Theorem 13. *As the* PAKE *is UC-Secure in presence of adaptive corruption under* SXDH*, the Oblivious Transfer scheme presented in Figure 2.17 is a secure 1-out-of-n Oblivious Transfer, with the same handling of corruptions as the* PAKE *protocol under* SXDH *and the* IND-CPA *security of the encryption scheme.*

On Another PAKE Scheme [KV09]

We are now going to show what happens when our general technique is used even on lattice-based PAKE. As an interesting example, we will apply it to the PAKE from [KV09] (see Figure 5.7). Lattices are drastically lacking a UC-PAKE, however we are going to apply our framework to this BPR secure scheme [BPR00].

We can see that this PAKE protocol is made of three flows. However the last flow is there to perform a key confirmation. The approximate SPHF used ensures that the two values $(\mathsf{tk}, \mathsf{tk}')$ completed are close-enough (in term of Hamming weight), so that the random key sk generated and coded into c can then be decoded.

Applying the Framework to Obtain a Secure OT Scheme

CRS generation:
$(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{param})$ for $\mathcal{E}$. $\mathsf{crs} = (\mathsf{pk})$
$\mathsf{flow}_1(\mathsf{pw}; r_1)$**:**

1. P_1 computes $C_1 = \mathsf{Encrypt}(\mathsf{pk}, \ell_1, \mathsf{pw}; r_1)$ where $\ell_1 = (\mathsf{sid}, \mathsf{ssid}, P_1, P_2)$
2. P_1 sends C_1 to P_2

$\mathsf{flow}_2(\mathsf{pw}; r_2)$**:**

1. P_2 computes $C_2 = \mathsf{Encrypt}(\mathsf{pk}, \epsilon, \mathsf{pw}; r_2)$.
2. P_2 generates $\mathsf{hk}_2 \leftarrow \mathsf{HashKG}(\mathfrak{L}_{\mathsf{pw},\ell_1})$, $\mathsf{hp}_2 \leftarrow \mathsf{ProjKG}(\mathsf{hk}_2, \mathfrak{L}_{\mathsf{pw},\ell_1}, C_2)$.
3. P_2 sends (C_2, hp_2) to P_1.

$\mathsf{flow}_3(\mathsf{pw}; r_2)$**:**

1. P_1 generates $\mathsf{hk}_1 \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_{\mathsf{pw},\epsilon})$, $\mathsf{hp}_1 \xleftarrow{\$} \mathsf{ProjKG}(\mathsf{hk}_1, \mathfrak{L}_{\mathsf{pw},\epsilon}, C_1)$.
2. P_1 computes $\mathsf{tk} = \mathsf{Hash}(\mathsf{hk}_1, \mathfrak{L}_{\mathsf{pw},\epsilon}, C_2) \cdot \mathsf{ProjHash}(\mathsf{hp}_2, \mathfrak{L}_{\mathsf{pw},\ell_1}, C_1, r_1)$
3. P_1 chooses $\mathsf{sk} \xleftarrow{\$} \{0,1\}^k$, computes $c = \mathsf{ECC}(\mathsf{sk})$
4. P_1 set $\Delta = \mathsf{tk} \oplus c$.
5. P_1 sends (Δ, hp_1) to P_2.

Key computing for P_2:

1. P_2 computes $\mathsf{tk} = \mathsf{Hash}(\mathsf{hk}_2, \mathfrak{L}_{\mathsf{pw},\ell_1}, C_1) \cdot \mathsf{ProjHash}(\mathsf{hp}_1, \mathfrak{L}_{\mathsf{pw},\ell_2}, C_2, r_2,)$
2. P_2 recover the key $\mathsf{sk} = \mathsf{ECC}^{-1}(\mathsf{tk} \oplus \Delta)$

Figure 5.7: Lattice-based PAKE from [KV09]

CRS generation:
$\mathsf{param}' \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}), (\mathsf{pk}', \mathsf{sk}') \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{param}')$
CRS: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}}), \mathsf{pk}'$.

Pre-flow :

1. $\mathcal{S}$ generates a LWE key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$
2. $\mathcal{S}$ stores sk, completely erase random coins used by $\mathsf{KeyGen}_{\mathrm{cpa}}$, and sends pk to $\mathcal{R}$

Index query on s:

1. $\mathcal{R}$ chooses a random J, computes $R \leftarrow F(J)$ and $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. $\mathcal{R}$ computes $C \leftarrow \mathsf{Encrypt}(\mathsf{pk}', \ell, s; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$
3. $\mathcal{R}$ stores r, R, completely erases the random coins used and sends C and c to $\mathcal{S}$.

Database input $(m_1, \ldots, m_n)$:

1. $\mathcal{S}$ computes $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and gets $R \leftarrow F(J)$
2. For $k = 1, \ldots, n$
   - $\mathcal{S}$ computes $\mathsf{hk}_k \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_{k,\ell})$, $\mathsf{hp}_k \leftarrow \mathsf{ProjKG}(\mathsf{hk}_k, \mathfrak{L}_{k,\ell}, C)$, $\mathsf{tk} \leftarrow \mathsf{Hash}(\mathsf{hk}_k, \mathfrak{L}_{k,\ell}, C)$,
   - $\mathcal{S}$ chooses $\mathsf{sk}_k \xleftarrow{\$} \{0,1\}^m$, sets $e_k = \mathsf{ECC}(\mathsf{sk}_k)$, and $\Delta_k = \mathsf{tk}_k \oplus e_k$
   - $\mathcal{S}$ computes $Q_k \leftarrow R \oplus \mathsf{sk}_k \oplus m_k$
3. $\mathcal{S}$ erases everything except $(\mathsf{hp}_k, Q_k, \Delta_k)_{k=1,\ldots,n}$ and sends it to $\mathcal{R}$

Data recovery:
Upon receiving $(\mathsf{hp}_t, Q_t)_{t=1,\ldots,k}$, $\mathcal{R}$ computes $\mathsf{tk}_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, \mathfrak{L}_{G(s),\ell}, C, r)$, $\mathsf{sk}_s = \mathsf{ECC}^{-1}(\Delta_s \oplus \mathsf{tk}_s)$, $m_s \leftarrow R \oplus \mathsf{sk}_s \oplus Q_s$, and erases everything besides m_s.

Figure 5.8: 1-out-of-n OT from Lattice-based PAKE

However, since with our framework the server does not need to commit to his password (the line index here), we can slightly modify the protocol and end up with the OT protocol exposed on Figure 5.8.

The security proof of this construction is a little beyond the point of this transformation, as the resulting scheme is not UC-Secure[14], however the BPR security of the PAKE ensures some minimum requirements.

As a participant in the PAKE cannot learn the other password, this ensures that even after seeing the receiver's flows, the server does not learn the line requested.

Similarly, since in case of password mismatch, a user cannot guess the key computed by the other one, this ensures that the user accesses at most one line.

[14]There is a problem to simulate an honest user. The Index query cannot be equivocated as the approximate SPHF Smoothness prevents the simulator to this circumvent this requirement.

In terms of efficiency, the resulting Oblivious Transfer ends up being slightly more efficient than both UC constructions from [GH08, BC15], but this comes at no surprise considering the security requirement gap. However this remains promising for the validity of our framework.

5.2.4 Very Efficient Oblivious Transfer from QA-NIZK

We now consider the instantiation from [JR15] in which the UC-PAKE allows for each party to hide his own password. As before, we first recall their protocol and then show how to use it in order to instantiate a new UC-secure Oblivious Transfer. This OT scheme ends up being the most efficient to date, as we show by comparing to the recent [BC16] in Table 5.11.

Interestingly this protocol does not use Smooth Projective Hash Function, but Quasi Adaptive Non-Interactive Zero Knowledge proofs [JR13]. It is presented in Figure 5.9.

A Quasi Adaptive Non-Interactive Zero Knowledge Argument is a NIZK (2.6) argument where the soundness property is computational but adaptive. Let us describe formally what is a Quasi Adaptive Non-Interactive Zero Knowledge Argument.

Definition 30 ((Labelled) Quasi Adaptive Non-Interactive Zero Knowledge Argument)**.** *Assuming a randomness distribution $\mathcal{D}_{\mathsf{param}}$ (for a public set of parameters* param*) and a set of languages $\{\mathfrak{L}_\rho\}_\rho$ parameterized by a randomness $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$ and associated with a relation $\mathcal{R}_\rho$ (meaning that a word x belongs to $\mathfrak{L}_\rho$ if and only if there exists a witness w such that $\mathcal{R}_\rho(x, w)$ holds), a* QA-NIZK *argument Π for this set of languages $\{\mathfrak{L}_\rho\}_\rho$ consists of five probabilistic polynomial time (PPT) algorithms $\Pi = (\mathsf{Gen}_{\mathsf{param}}, \mathsf{Gen}_{\mathsf{crs}}, \mathsf{Prove}, \mathsf{Sim}, \mathsf{Ver})$ defined as follows:*

- $\mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$ *returns the public parameters* param*.*
- $\mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$ *returns a common reference string* crs *and a trapdoor* tk*. We assume that* crs *contain the parameters* param*, a description of the language $\mathfrak{L}_\rho$ and a description of a label space $\mathcal{L}$.*
- Prove *takes as input* crs*, a label $\ell \in \mathcal{L}$, a word x of the language $\mathfrak{L}_\rho$ and a witness w corresponding to this word. If $(x, w) \notin \mathcal{R}_\rho$, it outputs failure. Otherwise, it outputs a proof π.*
- Ver *takes as input* crs*, a label ℓ, a word x and a proof π. It outputs a bit b (either 1 if the proof is correct, or 0 otherwise).*
- Sim *takes as input* crs*, a trapdoor* tk*, a label $\ell \in \mathcal{L}$ and a word x. It outputs a proof π for x (not necessarily in $\mathfrak{L}_\rho$) with respect to the label ℓ.*

These algorithms must satisfy the following properties

Perfect Completeness: *for all PPT adversary $\mathcal{A}$, all security parameter $\mathfrak{K}$, all public parameters* $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$*, all randomness $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$, all label $\ell \in \mathcal{L}$ and all $(x, w) \in \mathcal{R}_\rho$, the following holds:*

$$\Pr[(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \ell, x, w) : \mathsf{Ver}(\mathsf{crs}, \ell, x, \pi) = 1] = 1$$

CRS generation:
$g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, a, c, o, d, e, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_p, H \xleftarrow{\$} \mathcal{H}$
Set $A = g_1^a, D = g_1^d, E = g_1^e, U_1 = g_1^{u_1}, U_2 = g_1^{u_2} \in \mathbb{G}_1$
And $B = g_2^c, O = g_2^o, V_1 = g_2^{u_1 c - d - oa}, V_2 = g_2^{u_2 c - e} \in \mathbb{G}_2$
$\mathsf{crs} := (g_1, g_2, A, B, O, F, D, E, U_1, U_2, V_1, V_2, H)$

$\mathsf{flow}_b(\mathsf{pw}; (r_b, s_b))$**:**

1. P_b computes $R_b = g_1^{r_b}, S_b = \mathsf{pw} \cdot A^{r_b}, T_b = (D \cdot E^{h_b})^{r_b}, \hat{\rho}_b = B^{s_b}, W_b = (U_1 \cdot U_2^{h_b})^{r_b}$, where $h_b = H(\mathbf{sid}, \mathbf{ssid}, P_b, P_{\bar{b}}, R_1, S_1, \hat{\rho}_b)$
2. P_b erases r_b, sends $R_b, S_b, T_b, \hat{\rho}_b$ and keeps W_b.

Key computing:

1. P_b computes $h_{\bar{b}} = H(\mathbf{sid}, \mathbf{ssid}, P_{\bar{b}}, P_b, R_{\bar{b}}, S_{\bar{b}}, \hat{\rho}_{\bar{b}})$

   Checks that all elements are consistent, and computes:

   $\rho = g_2^{s_b}, \theta = O^{s_b}, \gamma = (V_1 \cdot V_2^{h_{\bar{b}}})^{s_b}$

   $\mathsf{sk} = e(T_{\bar{b}}, \rho) \cdot e(S_{\bar{b}}/\mathsf{pw}, \theta) \cdot e(R_{\bar{b}}, \gamma) \cdot e(W_b, \hat{\rho}_{\bar{b}})$

Figure 5.9: UC secure PAKE from [JR15]

Computational Adaptive Soundness: *For a given security parameter $\mathfrak{K}$, a scheme achieves computational adaptive soundness if for all PPT adversary $\mathcal{A}$, the probability*

$$\Pr[\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K}); \rho \leftarrow \mathcal{D}_{\mathsf{param}}; (\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho); (\ell, x, \pi) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{crs}, \rho) : x \notin \mathfrak{L}_\rho \wedge \mathsf{Ver}(\mathsf{crs}, \ell, x, \pi) = 1]$$

is negligible.

Perfect Zero-Knowledge: *for all $\mathfrak{K}$, all $\mathsf{param} \leftarrow \mathsf{Gen}_{\mathsf{param}}(\mathfrak{K})$, all $\rho \leftarrow \mathcal{D}_{\mathsf{param}}$, all $(\mathsf{crs}, \mathsf{tk}) \leftarrow \mathsf{Gen}_{\mathsf{crs}}(\mathsf{param}, \rho)$, all label $\ell \in \mathcal{L}$ and all $(x, w) \in \mathcal{R}_\rho$, the distributions $\mathsf{Prove}(\mathsf{crs}, \ell, x, w)$ and $\mathsf{Sim}(\mathsf{crs}, \mathsf{tk}, \ell, x)$ are the same.*

QA-NIZK-based PAKE

As explained in the original paper [JR15], the scheme is loosely-based on Quasi-Adaptive NIZK, in the sense that the flows use computation very close to what would be expected. In the figure 5.9, $\mathcal{H}$ is a family of hash functions.

5.2.5 Applying the Framework to Obtain a UC-Secure OT Scheme

Using the instantiation from [JR15] described in Figure 5.9, we apply our framework and directly obtain the protocol described in Figure 5.10.

The function G is assumed to map line numbers to group elements, and it should be collision-resistant and efficiently computable. One can use for example, the G function from Lindell in [Lin11].

CRS generation:
$g_1 \xleftarrow{\$} \mathbb{G}_1, g_2 \xleftarrow{\$} \mathbb{G}_2, a, c, o, d, e, u_1, u_2 \xleftarrow{\$} \mathbb{Z}_p, H \xleftarrow{\$} \mathcal{H}$
Sets $A = g_1^a, D = g_1^d, E = g_1^e, U_1 = g_1^{u_1}, U_2 = g_1^{u_2} \in \mathbb{G}_1$
And $B = g_2^c, O = g_2^o, V_1 = g_2^{u_1 c - d - oa}, V_2 = g_2^{u_2 c - e} \in \mathbb{G}_2$
$\mathsf{crs} := (g_1, g_2, A, B, O, F, D, E, U_1, U_2, V_1, V_2, H)$

Pre-Flow:

1. $\mathcal{S}$ generates $(\mathsf{pk} = f_1 = g_1^\alpha, \mathsf{sk} = \alpha \xleftarrow{\$} \mathbb{Z}_p)$
2. $\mathcal{S}$ erases random coins, keeps sk and sends pk.

Index query on i:

1. $\mathcal{R}$ picks $J \xleftarrow{\$} \mathbb{G}_1, s \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{mask} \leftarrow F(J)$ and $c = (f_1^s J, g_1^s)$
2. $\mathcal{R}$ picks $r \xleftarrow{\$} \mathbb{Z}_p$ computes $R = g_1^r, S = G(i) \cdot A^r, T = (D \cdot E^{h_{\mathcal{R},\mathcal{S}}})^r, W = (U_1 \cdot U_2^{h_{\mathcal{R},\mathcal{S}}})^r$, where $h_{\mathcal{R},\mathcal{S}} = H(\mathbf{sid}, \mathbf{ssid}, \mathcal{R}, \mathcal{S}, R, S, c)$
3. $\mathcal{R}$ erases r, s, J, sends $c, C = (R, S, T)$ and keeps W, mask.

Database answer:

1. $\mathcal{S}$ sets $J = c_1 / c_2^\alpha$, $\mathsf{mask} = F(J)$ and $h_{\mathcal{R},\mathcal{S}} = H(\mathbf{sid}, \mathbf{ssid}, \mathcal{R}, \mathcal{S}, R, S, c)$
2. For $k = 1, \ldots, n$:
   (a) Picks $s_k \xleftarrow{\$} \mathbb{Z}_p$
   (b) Sets $\rho_k = B^{s_k}, \rho'_k = g_2^{s_k}, \theta_k = O^{s_k}, \gamma_k = (V_1 \cdot V_2^{h_{\mathcal{R},\mathcal{S}}})^{s_k}$
   (c) Computes $K_{\mathcal{R},\mathcal{S},k} = e(T, \rho'_k) \cdot e(S/G(k), \theta_k) \cdot e(R, \gamma_k)$
   (d) Sets $Q_k = L_k \oplus \mathcal{H}(K_{\mathcal{R},\mathcal{S},k} \oplus \mathsf{mask})$
3. $\mathcal{S}$ sends all $(Q_k, \rho_k)_{k \in [\![1,n]\!]}$ and erases everything else.

Data recovery:
$\mathcal{R}$ computes $K_{\mathcal{R},\mathcal{S},i} = e(W, \rho_i)$, and recovers $L_i = Q_i \oplus \mathcal{H}(K_{\mathcal{R},\mathcal{S},i} \oplus \mathsf{mask})$

Figure 5.10: UC secure OT from QA-NIZK

Theorem 14. *As the* PAKE *was UC-Secure in presence of adaptive corruption under* SXDH*, and ElGamal is* IND-CPA *under* SXDH *too, the Oblivious Transfer scheme presented in Figure 5.10 is a secure 1-out-of-n Oblivious Transfer, with the same handling of corruptions as the* PAKE *protocol under* SXDH*.*

The PAKE scheme directly fits in the framework, hence the security is inherited from the initial PAKE proven secure in [JR15].

The same basic arguments apply: We first start by switching the crs so as to be able to simulate password openings, which will allow an honest receiver to commit to a dummy line value, and retroactively compute the opening value for any possible line s. On the other hand, an honest server will extract the committed line number, and provide dummy answers for all the other ones. If the receiver is honest, then the server will also send a dummy answer for the valid line, since the encrypted value S provides the extra degree of freedom to open to any line value L_k provided belatedly by the environment.

Efficiency Comparison We compare our efficient UC secure Oblivious Transfer with recent ones. This is depicted in figure 5.11. Note that in the case where there are many queries for the same database the adaptive OT from Section 4.3 will be more efficient.

Communication cost comparisons of various Elliptic Curve based OT schemes

Paper	Assumption	# Group elements	# Rounds
Adaptive Security (1-out-of-2)			
[ABB+13]	SXDH	$12\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + 2\mathbb{Z}_p$	3
[BC15][15]	DDH	$15\ \mathbb{G} + 2\ \mathbb{Z}_p$	3
[BC16]	SXDH	$12\ \mathbb{G}_1 + 4\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
This paper	SXDH	$6\ \mathbb{G}_1 + 2\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
Adaptive Security (1-out-of-n)			
[ABB+13]	SXDH	$\log n\ \mathbb{G}_1 + (n + 8\log n)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
[BC15][15]	DDH	$(n + 9\log n + 4)\ \mathbb{G} + 2n\ \mathbb{Z}_p$	3
[BC16]	SXDH	$4\ \mathbb{G}_1 + (4n + 4)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3
This paper	SXDH	$4\ \mathbb{G}_1 + (n + 2)\ \mathbb{G}_2 + n\ \mathbb{Z}_p$	3

Figure 5.11: Comparison to existing Oblivious Transfer

[15] It should be noted that the [BC15] OT does not rely on pairings. Classical implementations suggest that the elements in one of the groups in our scheme will be at least twice as big as those from the non-pairing curve, which would give roughly the same efficiency between both schemes in the 1-out-of-2 case, with an edge for the [BC15] construction in term of computational requirements. However, the scaling in their paper is then worse when increasing the number of lines (see the 1-out-of-n case).

Chapter 6

Conclusion

Conclusion In this thesis we explored a lot of possible applications to Identity-based Cryptography, creating new Identity-based primitives each time we needed them. Sometimes new kind of primitives like DIBE and sometimes new instantiation/transformation like our transformation from IBE to BIBE. Also we focused on Oblivious Transfer itself with and without Smooth Projective Hash Functions. They were useful to find an instantiation of OLBE and secure our adaptive version of the OT but too slow to create a very fast OT.

Open Problems and future Work There are different axes of research in this thesis which leads to different future works. I think that since a lot of this thesis is based on IBE any new IBE with better efficiency could impact most of it. Indeed better building blocks allow better transformations. For example if we could find a tightly-secure DIBE, this will increase the impact of our work and give a more practical scheme. In my opinion our transformation from DIBE to ABE could be a starting point for future works. First the DIBE scheme itself is new and can maybe find different uses. We already describe how it can instantiate some IBE's variant, it would be interesting to check every applications of these primitives and see if our efficient instantiation can solve practicality problems. Secondly the fact that the access structure of the ABE has to be a DNF formula is a bit restrictive, it would be great to generalize to any kind of access structure without increasing the communication cost that much e.g. staying linear in the number of attributes.

Bibliography

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Heidelberg, May 2010.

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013.

[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, Part II*, volume 4052 of *LNCS*, pages 300–311. Springer, Heidelberg, July 2006.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

[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*, volume 4734 of *LNCS*, pages 139–154. Springer, Heidelberg, September 2007.

[Att16] Nuttapong Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 591–623. Springer, Heidelberg, December 2016.

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer, Heidelberg, May 2004.

[BBC+13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, February / March 2013.

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 65–86. Springer, Heidelberg, June 2015.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016.

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 217–247. Springer, Heidelberg, December 2016.

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 579–598. Springer, Heidelberg, July 2017.

[BCGJ17] Olivier Blazy, Emmanuel Conchon, Paul Germouty, and Amandine Jambert. Efficient id-based designated verifier signature. *Interna-*

tional Conference on Availability, Reliability and Security, 12(44), 2017.

[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*, volume 4586 of *LNCS*, pages 274–292. Springer, Heidelberg, July 2007.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

[BFI+10] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 218–235. Springer, Heidelberg, June 2010.

[BFPV11] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, Heidelberg, March 2011.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, Heidelberg, March 2012.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCG+94] Richard Chang, Benny Chor, Oded Goldreich, Juris Hartmanis, Johan Hastad, Desh Ranjan, and Pankaj Rohatgi. The random oracle is false. *Journal of Computer and System Sciences*, 49(1):24–39, 1994.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.

[CDN09] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09*, pages 131–140. ACM Press, November 2009.

[CDVW12] Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 449–466. Springer, Heidelberg, May 2012.

[CFH+07] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 218–227. Springer, Heidelberg, November 2007.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 595–624. Springer, Heidelberg, April 2015.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May 2010.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, Heidelberg, April / May 2002.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013.

[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 *LNCS*, pages 360–363. Springer, Heidelberg, December 2001.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998.

[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, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[DG17] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 537–569. Springer, Heidelberg, August 2017.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[EMJ17] Nadia El Mrabet and Marc Joye. *Guide to pairing-based cryptography*. Chapman and Hall/CRC, 2017.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, Heidelberg, August 2006.

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, Heidelberg, December 2007.

[GH08] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, Heidelberg, December 2008.

[GHPT17] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 194–224. Springer, Heidelberg, August 2017.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[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.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, volume 2501 of *LNCS*, pages 548–566. Springer, Heidelberg, December 2002.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

[Gui13] Aurore Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 357–372. Springer, Heidelberg, June 2013.

[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.

[JR13] Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

[JR15] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to UC-PAKE and more. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 630–655. Springer, Heidelberg, November / December 2015.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Heidelberg, May 2005.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, December 2009.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 182–189. ACM, July 2003.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, Heidelberg, May 2011.

[LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 180–198. Springer, Heidelberg, August 2012.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal designated-verifier signatures. In Chi-Sung Laih, editor, *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 523–542. Springer, Heidelberg, November / December 2003.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Heidelberg, May 2005.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 165–179. Springer, Heidelberg, August 1997.

Abstract

We investigate the possibilities that **Identity-based Encryption** offers when used out of their original purpose. We manage to generalize a whole class of different identity-based encryption schemes into Downgradable Identity-based Encryptions. We found a generic way to construct Blind Identity-based Encryptions. These two works lead both to applications that are not a priori linked with IBE: Attribute-based Encryption from Downgradable IBE and Oblivious Transfer from Blind IBE. In the case of Affine IBE we manage to reduce the communication cost from linear to logarithmic. As application we also find a way to use Hierarchical IBE to construct a special type of signature called Identity-based Designated Verifier Signature. We continue the research out of the context of IBE's application with Oblivious Transfer. We manage to generalize the concept of Oblivious Transfer into a new protocol called Oblivious Language-based Envelope encompassing many kind of protocols. Finally, we construct Oblivious Transfer with a very different primitive called Password Authenticated Key Exchange. Surprisingly, with some optimizations this last transformation leads to a very efficient Oblivious Transfer Protocol. The Identity-based Encryption is our main basis of work, thus efficient instantiations of this primitive were the key of our own efficiency, thus we used the instantiation from Blazy et als at CRYPTO'14 which is efficient, tight secure and affine.

Résumé

Dans cette thèse nous étudions les possibilités que les **chiffrements basés sur l'identité** offrent quand ils sont utilisés dans un but différent qu'un simple chiffrement. Nous avons pu généraliser différents types de chiffrement basés sur l'identité en une nouvelle primitive nommé Downgradable Identity-based Encryption (DIBE). Nous avons trouvé un moyen générique de transformer de simple IBE en des IBE en blanc, dans le cas où l'IBE est affine nous rendons le coût de communication très faible (de linéaire à logarithmique). Ces deux primitives ont donné lieux à différentes applications : les chiffrements basés sur les attributs pour la première et le transfère inconscient pour la deuxième. Une autre application est l'utilisation d'IBE hiérarchiques pour créer des signatures à vérifieur désigné basées sur l'identité. Ensuite nous nous avons regardé le transfère inconscient seul et avons réussi à le généraliser en un nouveau protocole nommé Oblivious Language-based Envelope. Finalement, nous avons construit une transformation d'un protocole à un autre, d'un echange authentifié de clés par mot de passe nous avons construit un transfère inconscient. En prenant une instantiation particulère nous obtenons un protocole plus efficace que tous les précédents pour le même niveau de sécurité. La primitive chiffrement basé sur l'identité est notre outil principal pour réaliser nos constructions. Nous avons donc besoin d'une instantiation efficace de cette primitive. Nous avons utilisé celle de Blazy Kiltz et Pan à CRYPTO'14 qui est très efficace mais possède aussi une structure particulière dite affine.