On the achievability of white-box cryptography

Răzvan Roşie

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École normale supérieure

On the Achievability of White-Box Cryptography

Soutenue par
Răzvan Roşie
Le 28 mai 2019

Ecole doctorale n° 386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Michel FERREIRA ABDALLA Professeur, ENS	*Directeur de thèse*
Pierre-Alain FOUQUE Professeur, Université Rennes 1	*Rapporteur* *Président du jury*
Marc JOYE Docteur, OneSpan	*Rapporteur*
Tanja LANGE Professeur, TU Eindhoven	*Examinatrice*
Brice MINAUD Docteur, ENS	*Examinateur*
David NACCACHE Professeur, ENS	*Examinateur*
Pascal PAILLIER Docteur, CryptoExperts	*Examinateur*

ENS ÉCOLE NORMALE SUPÉRIEURE 1794 PSL RESEARCH UNIVERSITY PARIS

On the Achievability of White-Box Cryptography

Răzvan ROŞIE

Supervisor: Michel FERREIRA ABDALLA

Résumé

Cette thèse s'intéresse à la faisabilité des implémentations en boîte blanche de permutations pseudo-aléatoires sûres. Concrètement nous montrons comment un schéma de chiffrement fonctionnel à plusieurs entrées, qui satisfait une notion naturelle d'être à sens unique, est fondamental à la construction d'implémentations protégées contre les attaques d'extraction de clés. En outre, nous montrons comment réaliser de telles implémentations à partir de schémas de chiffrement fonctionnel existants. Bien que possédant des limitations, nous pensons que cette approche éclaire des questions sur la cryptographie en boîte blanche, peu représentée actuellement dans la littérature cryptographique.

Comme contribution indépendante possédant son intérêt propre, nous étendons la notion de robustesse cryptographique à des primitives variées. Sommairement, le chiffrement robuste garantit qu'un chiffré ne peut être lu au moyen de plusieurs clés. Des versions renforcées de cette notion protègent également des situations où l'adversaire génère les clés. Décrite tout d'abord dans le contexte de la cryptographie à clé publique, nous étendons les définitions aux contextes du chiffrement fonctionnel et à l'authentification. Enfin nous donnons des transformations simples mais génériques pour doter un schéma d'authentification (respectivement de chiffrement) de robustesse, tout en maintenant la sécurité du schéma d'origine.

Abstract

This thesis investigates the realizability of white-box implementations for secure pseudorandom permutations. Concretely, we show that multi-input functional encryption achieving a natural definition of one-wayness is instrumental in building implementations that are secure against key-extraction attacks. Furthermore, we show such implementations can be instantiated from existing functional-encryption constructions. Although impractical, we believe this proposal sheds some light over the field of white-box cryptography, currently largely overlooked in the cryptographic literature.

As a contribution of independent interest, we extend the notion of robustness to a larger set of primitives. Roughly speaking, robust encryption guarantees that a ciphertext cannot be decrypted under different keys. Strengthened variations of this notion should guarantee this happens even under adversarially generated keys. Initially formalized in a public-key context, we introduce compelling definitions for authentication and functional encryption schemes. Finally, we present simple, generic transforms that turn an authentication (respectively encryption) scheme into a robust one while maintaining the original scheme's security.

Contents

Chapter 1

Introduction

Cryptography stands as one of the cornerstones of modern daily life. Whether using a smartphone, PC or Mac, an increasing number of people depend on tools such as text-messaging, online banking, electronic mail or on documents received via various electronic applications. In the back end, all these user-friendly applications have to ensure (in theory) the privacy of the data they use, as well as the authenticity of the users accessing the data.

8 Bits of History. From a historical point of view, cryptography played a fundamental role in transmitting intelligence among friendly forces, without jeopardizing the secrecy of the *intel* even when it fell in the hands of enemies. From as early as the Romans' writings, we find out about trivial encryption methods used during wars (for instance Caesar's cipher [Sin99]), which we now call *monoalphabetic substitution* ciphers. During the Middle Age, the cryptographic methods became more elaborate, exhibiting properties generically described as *polyalphabetic substitution*. But it was not until the 20^{th} century and the birth of formal computational methods that cryptography became a formal, rigorous and mathematically-driven field. Alonzo Church, as the father of *lambda calculus* [Chu36], introduced the first model of computation, which turned out to be instrumental in the realization of modern computers. Even if the current architecture of computing devices follow the design by John von Neumann of a register machine with its own clock, arithmetic logic unit and registers, it is theoretically equivalent to lambda calculus. To strengthen the direct link between the evolution of computer science and cryptography, the very first computing devices (that had, almost exclusively, military purposes) were involved in attacking encrypted military communication. The usage of Colossus (United Kingdom) and ENIAC (United States) to break the Lorenz military cipher or to obtain the thermonuclear bomb were well-advertised. Their main purpose was speeding up the computations that otherwise should have been carried out by humans. No doubt this played a significant role in the war effort, nonetheless, the contributions of the people working at Bletchley Park, as well as the Polish "Biuro Szyfrów", who had the merit of cryptanalysing the early version of the German Enigma, should also be mentioned.

Up to the period and including the Second World War we relied on encrypting *alphanumeric* data through rotor machines, whose *ad hoc* designs are considered insecure with respect to current basic security notions. The study of information theory through probability theory

coincided with the inception of modern cryptography. Nowadays, the entire approach in cryptographic design has fundamentally changed. A significant advancement was proving that *one-time pad* encryption – adding random keys over the plaintexts – is perfectly secret as long as the keys are used once, making this method the primary way of securing sensitive communications, such as the *telephone lines* connecting opposing Cold War countries or communications between embassies and various centres. As expected, key-reuse in *one-time pad* had dreadful consequences: in the 1950s, a Soviet spy-ring operating within British intelligence, including Kim Philby, Anthony Blunt, Guy Burgess, John Cairncross and Donald Maclean was neutralized after a member reused the secret key of the one-time pad while communicating with his Soviet handler; thus, the British counter-intelligence was able to find his codename: "Homer". Reflecting on this, although perfectly secure, one can quickly pinpoint the major issue of one-time pads: the length of their keys should be at least as long as the length of the transmitted message. Thus, distributing *long keys* among multiple members becomes a *logistically difficult* task.

The cryptographic revolution produced in 1970s. As the key-distribution problem became more pressing, people thought of conceiving methods for "non-secret encryption", where the secret keys are no longer required for encryption. Two distinct groups were responsible for introducing what is now called public-key encryption: Whitfield Diffie and Martin Hellman, on the one hand, published their original results on how to exchange a cryptographic key in a public fashion [DH76], and on the other hand, British GCHQ[1], through James Ellis and Malcolm Williamson considered developing the same "non-secret" encryption paradigm [Sin99]. However, the latter work has been classified until the end of the 1990s, thus receiving little to no credit for their original ideas.

Secret Key and Public Key Cryptography. As one can easily deduce from our earlier exposition, symmetric-key cryptography assumes a pre-shared encryption key that is kept *secret* by the two parties involved, say Alice and Bob. On the contrary, public-key schemes assume the existence of a pair of public/secret keys. The interface of an *encryption procedure* requires a public key, a message and a *randomness* term, while the secret key is needed throughout the *decryption procedure*. It is not hard to see that any public-key encryption (PKE) scheme can be immediately turned into a secret-key scheme by keeping the public-key secret. Moreover, public-key encryption can be further generalized. In this work, we consider *functional encryption* as one of the most general encryption paradigms.

Functional Encryption. Functional encryption (FE) [ONe10b; BSW11] is one of the most appealing cryptographic primitives, as it offers "surgical" access over encrypted data. Traditionally, cryptographic schemes have been constructed around the "all-or-nothing" paradigm – the decryption either recovers the entire plaintext or returns nothing. This view is challenged in the functional encryption setting: a datum M is encrypted under a master key[2], while functional keys sk_f are issued for f in some supported class of functions $\mathcal{F}_\lambda$. The possessor of sk_f learns $f(M)$ from the encryption of M, and (ideally) nothing else on M. Originally proposed in the public-key setting, FE generalizes on a beautiful sequence of primitives, starting with public-key cryptography itself [DH76; RSA78; Pai99], continuing to identity-based encryption[3] [Sha84; BF01] and ending up with more advanced primitives such as attribute-based encryption [GPSW06; Wat11] or predicate encryption [KSW08].

[1] The Government Communications Headquarters, United Kingdom.
[2] The master encryption key may be *public* or *private*, depending on the setting.
[3] In IBE, public-keys are represented as memorable identifiers (e.g. phone numbers, email addresses).

Authentication Mechanisms. Authentication mechanisms are another major benefactor of the cryptographic explosion with a significant impact in the wild. In the symmetric key setting, the existence of *message authentication codes*, as the main authentication mechanism, proved to be fundamental in the realization of further primitives, such as authenticated encryption. In practice, at the heart of authentication primitives are the so-called *hash functions* that are able to produce *hard-to-invert* digests of arbitrary long messages. The public-key counterparts of message authentication codes are called *digital signature* schemes. A signer of a message uses its secret key in order to produce a (randomly looking) string that, associated with the message, can be verified under the signer's public key. Both message authentication codes, authenticated encryption or digital signature schemes became extensively used in practice and even standardized to be used in software libraries or hardware implementations.

1.1 Provable Security

Proofs are abstract *mathematical* concepts, used to establish a certain mathematical *postcondition*, starting from an original working hypothesis – the *precondition*[4] – via a sequence of sound steps. In the context of cryptography, the vague term of *provable security* is concerned with the theoretical or practical behaviours of cryptographic primitives in front of adversaries attempting to break specific properties. A major benefit is offering formal guarantees that specific primitives can achieve certain properties: for instance, one can require the outputs of an encryption scheme to "look random". Generally speaking, there are two major general problems with provable security:

1. *Proving security is done with respect to a specific scenario.*

   An inherent problem comes with the fact that certain aspects cannot be easily modelled: even if the *one-time pad* reaches *perfect confidentiality* in a cryptographic sense, it says nothing on its security when one of the users, for instance, a defector, hands-in his key to an adversary[5]. Thus, even if certain properties are achievable, (paranoid) scenarios in which the scheme does not guarantee confidentiality still exist. As a general rule, a system is as secure as its weakest link; and if such a link includes human factors, it might be easier to attack it in this way rather than to defeat the cryptographic tools. Nevertheless, security experiments capture realistic scenarios, and therefore having a scheme resisting to broad classes of attacks is undoubtedly desirable, even from a practical perspective, when compared to constructions lacking such properties.

2. *Proving security is often done by relying on problems that are conjectured, but not proven to be intractable.*

   The concept of a *reduction* [Kar72] is the foundation on which complexity theory is built. First and foremost, a reduction is an algorithm – a well-defined sequence of steps that can be carried out on a computational device. This algorithm transforms a given instance of a problem A into an instance of a problem B, where the reduction algorithm is usually running in polynomial time in its input length. Being able to show that a

[4] For instance an axiom, an already proven statement, or a conjecture.

[5] See, for instance, the case of John Anthony Walker, who offered the Soviet Union war-winning capabilities against the United States in the mid-1980s'. Walker provided his Soviet handlers with cryptographic keys such that they were able to locate the US nuclear ballistic submarine fleet.

particular occurrence of a problem is reducible to the one of another problem allows for constructing a *hierarchy* of problems. One must distinguish here between the *worst* and *average* case complexity.

In cryptography, we reuse the same idea rooted in complexity theory. By beginning with a long-studied problem believed to be hard, one attempts to show that a particular security notion is achieved. In some sense, provable security is a like snake oil: ironically, the security of construction relies, in the end, on some unverified working hypothesis that has to be trusted.

A major open problem in computer science is proving or disproving if polynomial algorithms for a specific set of problems, denoted as NP-Complete problems exist. In cryptography, very often we rely on a class of *intermediate* NP problems (NPI) for which DTM (deterministic Turing machine) solvers running in *sub-exponential* time exist, nonetheless, it still remains an open problem to find solvers running in polynomial time. Throughout this work, it will be very often the case that proving that a scheme is "secure" concerning some scenario becomes equivalent with proving that an adversary breaks a specific NPI problem for which there is no known solver.

Bridging theory to practice. Algorithms are abstract descriptions, which have to be translated into programs to be executed by computing devices. In practice, it is often the case that the implementation itself leaks precious information on the secret values used by the cryptographic primitives. For instance, by tracing the power consumption or the memory-access patterns made by a program, an eavesdropper can mount key-extraction attacks. Such scenarios assume that cryptographic programs containing sensitive information are executed in adversarial environments (i.e. the adversary is given an executable). Thus, another dimension one has to take into account are the types of adversaries considered. From a theoretical point of view, it matters if the running time of an adversary is bounded to a polynomial of the input length or if it is unbounded in *time* or *space*. Usually, an adversary is modelled as an algorithm given "black-box" access to procedures interacting with secret-keys, such as decryption or key-derivation. Practical adversaries are "white-box", by literally inspecting the internal working of a procedure. This existing gap remains, and bothers both practitioners and theoreticians.

1.1.1 Key-Extraction Attacks

As stated above, in the realm of cryptography, there has always been a significant gap between theory and practice. In this work, we plan to make a step forward towards bridging the theoretical security notions behind the abstract algorithms to the concrete security of their implementations. Our motivation is twofold: on the one hand, we point out that most of the existing security notions treat schemes with kid gloves: many constructions shown secure in the *standard* or *random oracle* models, targeting enhanced security guarantees or practical efficiency are deployed in the wild, but fail to guarantee their claims. This happens mainly because proving schemes is done in a *black-box model*, while the real-life abounds in *white-box* adversaries, able to exploit the peculiarities of an implementation. On the other hand, practitioners developed heuristic methods that may behave appropriately when implemented, but for which there is no provable security. Thus, we think that being able to describe a complete (and proven) chain of theoretical relations that ensures security against white-box adversaries is of great interest for both theoreticians and practitioners.

White-Box Cryptography. White-box cryptography (WBC), introduced by the seminal paper of Chow, Eisen, Johnson and van Oorschotin [CEJv03], captures in an informal manner such adversarial capabilities. However, it was not until the work of Delerablee et al. [DLPR14] when rigorous (and simple) definitions were introduced. The central notion, dubbed *unbreakability*, assumes an adversary interacts with an "implementation" of a cryptographic primitive, which contains an embedded key, and has to extract the key. Although important, we point out that such a definition does not rule out an adversary partially learning the key. However, we assume that if this happens, either the adversary continues in its attempt to recover the key in its entirety or gives up. In the spirit of the original definitions, the model we consider assumes correct key generation and no prior adversarial knowledge on the key (either partial knowledge or complete).

WBC and Obfuscation. Software obfuscation can be stated as the problem of creating functionally equivalent, but unintelligible programs. Unfortunately, general *virtual black-box obfuscation* has been proven impossible for general circuits [BGI+01]. A relaxation of the original notion – *indistinguishability obfuscation* (iO) – may still be possible to achieve and has become a major open problem attracting numerous research [GGH+13; CLT15; SW14]. A clear overview of the problems connected to iO is given in [Hor15]. In some respects, obfuscation shares some similarities with WBC (which aims at hiding a secret key into a circuit). However, iO does not straightly imply white-box unbreakability (UBK) in the sense that applying an iO compiler to an encryption program/circuit does not make it unbreakable (*i.e.* there is no guarantee extracting the key from the resulting program is difficult). The question of whether UBK could be obtained from iO is still open up to now. From a theoretical perspective, a significant gap between iO and UBK arises from the fact that the former is intrinsically an *indistinguishability* notion, while the latter is in fact modelled as a *computational* game.

Motivated by the lack of results in the field of white-box crypto, we leverage the power of functional encryption for general circuits in order to inspect whether *unbreakability* can be achieved through theoretical means. We explain in high-level the technique we employ towards obtaining a UBK-secure implementation.

WBC and Functional Encryption. Consider functional encryption schemes supporting circuits that compute one-way functions of the form $f : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$. If there exists a way to apply f over an input space consisting of $\{K||M \ : \ K \leftarrow_{\$} \mathcal{K}, M \in \mathcal{M}\}$, one would decrypt to $f(K, M)$. We note that such a setting is immediately enabled by multi-input functional encryption (MIFE) – the public-key setting. Moreover, we observe that if MIFE encryption is one-way in the presence of a functional key – meaning that once a plaintext has been encrypted, it is difficult to retrieve it even in the presence of a (single) functional key – the MIFE scheme is a good candidate for constructing a UBK compiler.

As an alternative, one may think about the decryption algorithm as taking an extra, *auxiliary input* – corresponding to M – such that $\mathsf{Dec}(\mathsf{sk}_f, C_K, M) = f(K, M)$, where C_K stands for the encryption of f's key. We call this notion as *functional encryption with auxiliary inputs* (FEAI).

1.2 Our Results

The main contributions presented herein can be regrouped over two main domains. The first stream of work contains concepts connected to the cryptographic notion of *robustness*. The results shown have already been published by the author in [FOR17] and [GNR19]. The second part is rooted in a sequence of results aiming at achieving *white-box* secure schemes. Most of the results are presented in one article currently under submission [GPR+19].

1.2.1 Robustness for Cryptographic Primitives

Robustness is a notion often tacitly assumed while working with encrypted data. Roughly speaking, it states that a ciphertext cannot be decrypted under different keys. Initially formalized in a public-key context, it has been further extended to key-encapsulation mechanisms, and more recently, to pseudorandom functions, message authentication codes and authenticated encryption.

1.2.1.1 Robustness for Symmetric Primitives

In [FOR17], Farshim, Orlandi and Roşie study the security of symmetric primitives under the *incorrect* usage of keys. Roughly speaking, a *key-robust* scheme does not output ciphertexts/tags that are valid with respect to distinct keys. Key-robustness is a notion that is often tacitly expected/assumed in protocol design – as is the case with an anonymous auction, oblivious transfer, or public-key encryption. The authors formalize simple, yet strong definitions of key robustness for authenticated-encryption, message-authentication codes and pseudorandom functions (PRFs). It is shown that standard notions (such as AE or PRF security) guarantee a basic level of key-robustness under honestly generated keys, but fail to imply key-robustness under adversarially generated (or known) keys. Robust encryption and MACs compose well through generic composition, having robust PRFs as the main primitive used in building robust schemes. Standard hash functions are expected to satisfy key-robustness and PRF security and hence suffice for practical instantiations. [FOR17] provides further theoretical justifications (in the standard model) by constructing robust PRFs from (left-and-right) collision-resistant pseudorandom generators (PRGs).

1.2.1.2 Robustness Extended to Further Public Primitives

In [GNR19], Géraud, Naccache and Roşie motivate the importance of establishing robustness guarantees for digital signatures (a signature must not verify under multiple keys), as well as for functional encryption schemes, even under adversarially generated, but well-formed keys. Scenarios that can result in attacks against existing constructions (such as the Boneh–Boyen signature scheme or a simple bounded-norm inner-product functional encryption scheme) if robustness fails are presented.

Furthermore, the work shows there exist simple, generic transforms that convert a scheme into a functionally equivalent but robust one, preserving, in particular, the original scheme's guarantees.

1.2.2 Towards Achieving White-Box Cryptography

White-box cryptography is the final frontier in modeling real-world adversaries against cryptographic schemes. In rough terms, the attacker is provided with an implementation of a particular scheme, and he/she is asked to extract the secret key embedded in the implementation using every possible means. We say that a construction achieves *unbreakability* (UBK) if no adversary succeeds in extracting the key with a significant advantage. Despite its overwhelming importance in the real-world deployment of cryptographic schemes, no provably-secure constructions are known to this date. Even the recent developments in the field of *indistinguishability obfuscation* (iO) have not provided benefits to white-box cryptography. As a matter of fact, it is still unknown whether iO can be used to achieve white-box unbreakability for a given encryption scheme. Meanwhile, the community focused on the development of heuristic white-box implementations, out of which *none* resisted to subsequent attacks. In the recent "WhibOx" challenge of CHES 2017 (appealing for submissions of white-box AES implementations), *none* of the candidates managed to resist the attacks from the cryptographic community, which gives a (dramatic) panorama of the *status quo* on white-box cryptography.

1.2.2.1 Is UBK-security achievable?

Motivated by the lack of theoretical results in this field, we focus on achieving unbreakability for encryption schemes. The main contributions are under submission [GPR+19] and can be summarized as follows:

We give a positive answer to the question of obtaining white-box implementations for block-ciphers that are secure against adversaries targeting key-extraction (UBK-security). Although relying on strong assumptions, to the best of our knowledge, this is the first time when the problem has been rigorously studied and answered. To this end, we formalize one-wayness for functional encryption, by introducing simple but powerful definitions for multi-input functional encryption schemes – denoted OW-MIFE– in both the public and private-key settings. We show both settings are sufficient to achieve UBK-secure implementation for a pseudorandom permutation f.

Then, we look into the power conferred by multi-input functional encryption in the private-key setting. Specifically, we investigate the generic transform introduced by Brakerski, Komargodski and Segev in [BKS16] (from now on the "BKS" transform). This builds a n-input MIFE scheme on top of a single input FE scheme for general circuits which is function-hiding. We prove the BKS transform enjoys one-wayness assuming the original single input scheme is one-way. Furthermore, we argue that such single input schemes can be instantiated from standard assumptions, by referring to the construction of Goldwasser *et al.* [GKP+13].

By making use of known results in the realm of multi-input functional encryption, we show that assuming the existence of indistinguishability obfuscation and of one-way functions, an UBK implementation can be achieved, via the transform by Goyal, Jain and O'Neill [GJO16]. As a contribution of independent interest, we propose the concept of functional encryption with auxiliary inputs (FEAI), as well as its indistinguishability and one-wayness security notions. We show IND-FEAI schemes are sufficient to obtain indistinguishability obfuscation. We regard this as an alternative potential path for obtaining iO, a problem of independent interest. We also show that one-way FEAI (OW-FEAI) enables to achieve UBK-security.

1.3 Other Contributions

Several other minor contributions, published or in submission, were left aside and not included in this thesis. Nonetheless, we believe they may present interest from both theoretical and practical points of view, and we mention them briefly in what follows:

1.3.1 Adaptive-Secure VRFs with Shorter Keys from Static Assumptions

Verifiable random functions are pseudorandom functions producing publicly verifiable proofs for their outputs, allowing for efficient checks of the correctness of their computation. In [Ros18], the author first introduces a new computational hypothesis, the n-Eigen-Value assumption – which can be seen as a relaxation of the $\mathcal{U}_n$-MDDH assumption – and proves its equivalence with the n-Rank assumption. Based on the newly introduced computational hypothesis, the core of a verifiable random function having an exponentially large input space and reaching adaptive security under a static assumption is built. The final construction achieves shorter public and secret keys compared to the existing schemes reaching the same properties. The results of this work have been published in the proceedings of CANS 2018.

1.3.2 Compressing Transactional Ledgers

Banks and blockchains need to keep track of an ever-increasing list of transactions between the accounts owned by their users. However, as time goes by, many of these transactions can be safely *forgotten*, in the sense that purging a set of transactions that compensate each other does not impact the network's *semantic meaning*, i.e. the vector b of amounts representing the balances of users at a given point in time t.

[GNR17] introduces the notion of *nilcatenation* – a collection of past transaction vectors having no effect on b. Removing these transactions yields a smaller, but equivalent set of transactions. Motivated by the computational and analytic benefits obtained from more compact representations of numerical data, Géraud, Naccache and Roşie formalized the problem of finding nilcatenations, and propose detection algorithms. Among the suggested applications are decoupling of centralized and distributed database or even to lower the burden of large nodes maintaining the increasing blockchains of financial transactions. The nilcatenation detection algorithm can be seen as proof of useful work, as the periodic removal of nilcatenations keeps the ledger's size as small as possible. The results in this work was published in the proceedings of SecureComm 2017.

1.3.3 Further Algorithms for Simple, SSP Related Problems

The subset-sum problem (SSP) was among the first to be proven NP-Complete; it underwent thorough analysis and has been used in numerous applications, such as public-key encryption, signature schemes, or pseudorandom number generation. In most practical scenarios, the security of such constructions depends on the fastest-known algorithms to solve "randomized" instances of the SSP.

However, natural variants of the SSP, such as the subset-product problem (SPP) or the multi-dimensional SSP (MDSSP), did not undergo such scrutiny, despite them appearing in some contexts of practical interest.

In this work, the authors introduce efficient polynomial reduction of these variants to the SSP, which results in algorithms with improved running time, under the assumption that

the SSP solver has a notable property that is called "solution equiprobability". In particular, relying on (1) an equiprobable SSP solver running in $O(2^{n \cdot e + o(1)})$, returning solutions with probability $1/r$ and (2) on a hypergraph partitioner running in time $O(h)$ and producing partitions of maximal size ℓ, the authors give $O(r^\ell \cdot 2^{\ell \cdot e + o(1)} + h)$-time new algorithms for the *sparse* multiple SSP, from which an efficient subset-product solver is then constructed. An extended abstract of this result was included in AQIS 2018.

1.4 Organization

This dissertation is organized as follows. We begin by introducing the notations and the main primitives we use in Chapter 2. These definitions serve as the basis for the future chapters. In Chapters 3 and 4 we study the security notion of *robustness* in the context of encryption and authentication. For encryption primitives, we extend the already existing security definition for the public-key setting to a broader, functional setting. The authentication part studies similar guarantees for digital signatures and authenticated encryption. In Chapter 5, we transit to the study of security notions related to the implementation of schemes. The central notion of white-box cryptography is under scrutiny, showing that under specific cryptographic assumptions, it can be achieved in front of adversaries modelled as Turing machines. Finally, in Chapter 6, the notion of functional encryption with auxiliary inputs is formalized and linked to the concepts in the previous chapter. The concluding remarks and several open problems are presented in Chapter 7.

Chapter 2

Preliminaries

This chapter introduces the notations to be used herein, continuing with a presentation of relevant computational hardness assumptions and of standard primitives that represent the workhorse for the following chapters. We present only the main cryptographic concepts while postponing variations of them to the forthcoming chapters.

2.1 General Notations and Concepts

In this section, we introduce the main notations that shall be used to describe several mathematical and algorithmic concepts.

2.1.1 Mathematical Notations

Integers and Sets. We denote by $\mathbb{N}$ the set of natural numbers and by $\mathbb{Z}$ the set of integers. We write $\mathbb{N}^*$ and $\mathbb{Z}^*$ as $\mathbb{N} \setminus \{0\}$ and $\mathbb{Z} \setminus \{0\}$ respectively. For a finite set S, we denote its cardinality by $|S|$. From now on the symbol ":=" is to be used for defining new terms. We define $[k] := \{1, \ldots, k\}$ as the set of first k positive integers.

Matrices and Vectors. Variables in bold capital letters stand for matrices (e.g. $\mathbf{M}$) while bold lowercase letters represent vectors (e.g. $\mathbf{u}$). A subscript i on a vector $\mathbf{u}$ (e.g. $\mathbf{u}_i$) stands for the i-th component of the vector. An analogue convention is used for matrices. We abuse notation and by $[a]$ we also denote the "encoding of an element" with respect to some algebraic structure, and through $[\mathbf{M}]$ and $[\mathbf{u}]$, we denote the encodings of a matrix, respectively vector.

Groups, Rings and Modular Arithmetic. A *group* (algebraic structure) $\mathbb{G} = (S, \star)$ is defined as a set S and a law of composition $\star$ that takes two elements x, y from S and produces an element $x \star y$ also in S. A group satisfies the closure, associativity, neutral element, and inverse element definitions. We call Abelian a group that satisfies, also, commutativity. A *semigroup* $(S, \otimes)$ satisfies the closure and associativity definitions, while a *monoid* is semigroup equipped with a neutral element.
A *ring* (algebraic structure) $\mathcal{R} = (S, \oplus, \star)$ is defined with respect to two operations, which will be called *addition* and *multiplication*. We require $(S, \oplus)$ to be an Abelian group and $(S, \star)$ to be a monoid, while multiplication $\star$ to be distributive with respect to addition $\oplus$.
A *field* (algebraic structure) $\mathcal{F} = (S, \oplus, \star)$ is defined for two operations: called *addition* and *multiplication*. We require $(S, \oplus)$ to be an Abelian group having the identity element denoted by 0. $(S \setminus \{0\}, \star)$ forms an Abelian group. The multiplication $\star$ is distributive concerning addition $\oplus$.

Cyclic Groups. A group is called cyclic if it is finite and all elements can be generated from a single one, called the *group generator*. We usually denote the generator of a cyclic group by g. A cyclic group is Abelian.

Modular Arithmetic. For an integer n, we denote by $(\mathbb{Z}_n, +, \cdot)$ or $\mathbb{Z}/n\mathbb{Z}$ – or simply $\mathbb{Z}_n$ – the ring of integers modulo n. We represent the elements of a ring by their representatives $\{0, 1, 2, \ldots, n-1\}$. Observe that whenever n is a *prime* number, then $(\mathbb{Z}_n, +, \cdot)$ is a finite field.

Negligibility. In the cryptographic parlance, we often refer to *negligible functions*. What we mean through this terminology is that a function vanishes faster than any positive polynomial function. Using our notations, a real-valued function Negl is negligible if it is upper bounded by the inverse of any polynomial P: that is, there exists a rank N_0 such that for any $n > N_0$ we have $\text{Negl}(n) < 1/P(n)$. We state that an event occurs with overwhelming probability if its probability is $1 - \text{Negl}$. We denote the set of all negligible functions by Negl.

PROBABILITIES. A random variable $X : \Omega \to E$ is defined as a map between a set of possible outcomes Ω and a measurable space E. We denote by $\Pr[X = x]$ the probability the random variable X takes the value x. A *probability distribution* provides the probabilities of occurrence of an event. The *uniform distribution* over some set S, assigns the same probability to any element x in the set: $\Pr[X = x] = 1/|S|$.

ASSIGNMENTS. $s \leftarrow a$ stands for assigning the value a to variable s. We denote the action of sampling an element x *uniformly at random* from a set S by $x \leftarrow\!\!\$\ S$. When another, non-uniform distribution χ is to be used, we write $x \leftarrow_\chi S$.

BITSTRINGS AND LISTS. We denote an ordered list L of n elements by $(a_1, \ldots, a_n)$. For a real element $r \in \mathbb{R}$, we write $\lfloor r \rfloor$ for its integer part while defining $\lceil r \rceil := \lfloor r \rfloor + 1$. A bitstring is an element taken from a finite subset of $\{0,1\}^*$. We denote the binary representation (*bitlength*) of a positive integer n the following quantity $\lceil \log_2(n) \rceil$.

2.1.2 Algorithms and Turing Machines

According to [CLRS09], an algorithm is a sequence of steps taking some value denoted the *input* and produces *output* values. In this work, we regard algorithms as equivalent to Turing machines. As a general convention, we assume an algorithm to be randomized unless stated otherwise.

TURING MACHINES. A Turing machine is an abstract model of computation. As described in the original paper of [Tur37], a Turing machine consists of an infinite tape, a special set of symbols including a separator, starting and stopping symbols. A special reader tool is to be used, as well as a finite set of instructions that will let the machine in a finite set of states.

For cryptographic applications, we are mostly interested in algorithms whose running time is bounded by a polynomial in the total length of the input. We denote the security parameter by $\lambda \in \mathbb{N}^*$ and we assume it is implicitly given to all algorithms in the unary representation 1^λ.

TIME AND SPACE COMPLEXITY. The two main physical quantities in connection to the execution of an algorithm are the *time* and *space*. We will denote by the time complexity of an algorithm the total number of steps executed by the Turing machines while running the algorithm. The space complexity – or the necessary space to run the program description – is defined as the length of the used tape of the Turing Machine. In terms of notation, PPT is often used and stands for a probabilistic algorithm running in *polynomial-time* in the total length of its input(s).

FUNCTIONS AND ASYMPTOTIC BEHAVIOUR. Many times, computing the exact time/space complexity of an algorithm is a complex process. Instead, approximations are preferred. We say that a function $f : \mathcal{X} \to \mathcal{Y}$, $f(n) \in O(g(n))$ if $\exists n_0, c$ such that $\forall n > n_0$ we have that $f(n) \geq c \cdot g(n)$. Similarly, we say that $f(n) \in \omega(g(n))$ if there exists n_0, c such that $\forall n > n_0$ it is the case that $f(n) \leq c \cdot g(n)$. Finally, we say that f and g have the same behaviour and write $f(n) \in \Theta(g(n))$ if exists c_1, c_2, n_0 such that $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$, for all $n > n_0$.

2.1.3 A Toolkit for Proofs in Cryptography

We present here basic cryptographic concepts that shall be used in further definitions and forthcoming sections.

Security Parameters. Intuitively, in cryptography, by security parameter, we usually mean the length of some secret that provides guarantees in hardness of breaking a specific property. Formally, the security parameter is associated with the input length for the Turing machine that models a specific functionality.

Oracles. Oracles are fundamental tools in theoretical computer science. When we say that one is *given oracle* access to a function f, we mean that for any provided input x, one learns only the corresponding output $f(x)$, being agnostic on the way the internal computations of the function have been carried out. Oracles can be queried multiple times, even for the same input. We denote by $\mathcal{A}^{\mathcal{O}}$ that algorithm $\mathcal{A}$ is given access to oracle $\mathcal{O}$.

Security Experiments. A security experiment (vulgarized as a *game*) is intended to capture a realistic scenario in a formal manner. The game is, in fact, a randomized algorithm, that takes as input the security parameter, usually initializes a scheme and provides a challenging problem to an adversary (again, seen as a Turing machine). Running the experiment can be done multiple times. The algorithm modelling the game returns an output, indicating the adversary succeeded or not in winning the given challenge.

Modelling Adversaries. We usually write $\mathcal{A}$ to denote an adversary, seen as a PPT algorithm. Given a randomized algorithm $\mathcal{A}$, we denote the action of running $\mathcal{A}$ on input(s) $(1^\lambda, x_1, \dots)$ with uniform random coins r and assigning the output(s) to $(y_1, \dots)$ by $(y_1, \dots) \leftarrow \mathcal{A}(1^\lambda, x_1, \dots ; r)$. We recall the definitions for several standard cryptographic concepts related to adversaries.

Success Probability and Advantage. The distributions of the outputs returned by the algorithm modelled by the security experiment define the *advantage* of an adversary in winning the game. Assume that for a specific construction $\mathbf{C}$ there exists an adversary $\mathcal{A}$ against property $\mathbf{P}$. Acting as a *distinguisher*, $\mathcal{A}$ is given distribution $\mathcal{D}_0$ or distribution $\mathcal{D}_1$. We define the advantage as $\mathsf{Adv}^{\mathbf{P}}_{\mathcal{A},\mathbf{C}}(\lambda) \coloneqq \left| \Pr[1 \leftarrow \mathcal{A}(\mathcal{D}_0)] - \Pr[1 \leftarrow \mathcal{A}(D_1)] \right|$.

Perfect, Statistical and Computational Indistinguishability. Let $\mathcal{D}_0, \mathcal{D}_1$ be two probabilistic distributions defined over the same support space Ω. The *statistical distance* between the two distributions is defined as $SD(\mathcal{D}_0, \mathcal{D}_1) \coloneqq \frac{1}{2} \cdot \sum_{x \in \Omega} \left| \mathcal{D}_0(x) - \mathcal{D}_1(x) \right|$. We say that two distributions $\mathcal{D}_0, \mathcal{D}_1$ are ϵ-close if the $SD(\mathcal{D}_0, \mathcal{D}_1) \leq \epsilon$. We say that two distributions are perfectly indistinguishable if the statistical distance between the two is zero. We note that two distributions are statistically close even in front of computationally unbounded adversaries.

Many times, it suffices to assume that realistic adversaries' runtime is fixed, and therefore, statistical indistinguishability can be relaxed to a notion where two probability distributions are indistinguishable in front of a *computational* adversary. The indistinguishability condition stays as in the previous case, under the restriction that $\mathcal{A}$ is now a PPT adversary.

Hybrid Arguments in Cryptographic Security Proofs. As stated in the introduction, the theory of provable security is built on the assumptions that winning a security experiment is equivalent to breaking a specific problem believed to be hard to solve. However, many times, such direct reductions are infeasible, and one has to go through a slightly more convoluted argument, generically called as a *hybrid argument*. Suppose we want to show that $\mathcal{D}_0$ is indistinguishable from $\mathcal{D}_n$. One can do so by stepping through a sequence of hybrid experiments $H_1, \dots, H_n$ where the output distributions of each experiment can be proven to be indistinguishable from the output of the previous hybrid. Assuming the output

distribution of H_1 is $\mathcal{D}_1$ and of H_n is $\mathcal{D}_n$, via the *union bound* we get an argument for establishing indistinguishability.

Black-Box Model vs White-Box Model Security Models. Traditionally, security games were introduced in the so-called black-box model. Here, we model an adversary via a PPT Turing machine that interacts with oracles representing the encryption, decryption, key-derivation *et al.* procedures. In some sense, this model is idealized, as it assumes the adversary is entirely agnostic to what happens when a message is, for instance, encrypted.

In the real world, the adversary may be able to extract more information rather than just talking to an oracle. Nevertheless, we still model adversaries as Turing machines, in some sense being able to theoretically model the leakage functions for a particular implementation of an algorithm.

2.2 Cryptographic Primitives

2.2.1 Hash Functions and the Random Oracle Model

A hash function (see for instance the definition in Katz and Lindell [LK14]) is a keyed cryptographic primitive that takes as input arbitrary long messages and outputs message digests (usually) of a fixed length. Informally, we desire hardness in recovering the original message given its message digest, as well as hardness in finding messages that produce colliding digests.

Definition 2.1. *A hash function* $\mathsf{H} : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^l$ *is a deterministic Turing machine that, given a fixed, publicly available, and uniformly at random sampled key* K*, takes as input a message* M *of arbitrary length, and returns* $h \leftarrow \mathsf{H}_K(M)$*.*

We say that a hash function achieves preimage resistance if for any PPT *adversary* $\mathcal{A}$ *we have that:*

$$\mathsf{Adv}^{\mathrm{pr}}_{\mathcal{A},\mathsf{H}_K}(\lambda) \coloneqq \Pr\left[y = \mathsf{H}_K(x') \;\middle|\; \begin{array}{l} K \leftarrow\!\!\$\ \mathcal{K} \wedge x \leftarrow\!\!\$\ \{0,1\}^* \wedge \\ y \leftarrow \mathsf{H}_K(x) \wedge x' \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, \mathsf{H}_K, y) \end{array} \right] \in \textsc{Negl}(\lambda)\ .$$

Similarly, a hash function is collision-resistant if:

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathcal{A},\mathsf{H}_K}(\lambda) \coloneqq \Pr\left[\mathsf{H}_K(x) = \mathsf{H}_K(y) \;\middle|\; K \leftarrow\!\!\$\ \mathcal{K} \wedge (x,y) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, \mathsf{H}_K) \right] \in \textsc{Negl}(\lambda)\ .$$

Hash functions are valuable cryptographic objects. Reasons are twofold: first, when it comes to theory, it is always valuable to assume that a well-designed hash construction is close enough to an idealized *random oracle* [BR93], that perfectly emulates the uniform distribution over $\{0,1\}^l$. Such a reasonable assumption usually allows for clear ways of building proofs rather than looking into the convoluted mathematical structure behind it.

Second, when it comes to practical cryptographic applications, the existing constructions based on the theory of boolean functions proved to be fast enough to satisfy the software and hardware requirements.

2.2.2 Pseudorandom Generators

A pseudorandom generator PRG with domain $\mathcal{D}$ and range $\mathcal{R}$ is a deterministic algorithm that on input a point $x \in \mathcal{D}$ outputs a value $y \in \mathcal{R}$. We define the advantage of an adversary

$\mathcal{A}$ against PRG as

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{PRG}^{\mathcal{A}}_{\mathsf{PRG}}(\lambda) = 1\right] - 1 \ ,$$

where the game $\mathrm{PRG}^{\mathcal{A}}_{\mathsf{PRG}}(\lambda)$ is shown in Figure 2.1 (left). A PRG is secure if the above advantage function is negligible for every PPT adversary $\mathcal{A}$. In what follows, we assume $\mathcal{D}$ and $\mathcal{R}$ come with algorithms for sampling elements, which by slight abuse of notation we denote by $\mathcal{D}(1^\lambda)$ and $\mathcal{R}(1^\lambda)$. We allow for arbitrary domain and range in this definition to allow for the analysis of our constructions later on.

$\underline{\mathrm{PRG}^{\mathcal{A}}_{\mathsf{PRG}}(\lambda)}$:	$\underline{\mathrm{PRF}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda)}$:	$\underline{\text{Proc. } \textsc{Eval}(M)}$:
$b \leftarrow\!\!\$\ \{0,1\}$	$b \leftarrow\!\!\$\ \{0,1\}$; $\mathrm{L} \leftarrow \emptyset$	if $M \in \mathrm{L}$ then return $\perp$
$x \leftarrow\!\!\$\ \mathcal{D}(1^\lambda)$; $y \leftarrow \mathsf{PRG}(x)$	$K \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$	$T \leftarrow \mathsf{PRF}(K, M)$
if $b = 0$ then $y \leftarrow\!\!\$\ \mathcal{R}(1^\lambda)$	$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Eval}}(1^\lambda)$	if $b = 0$ then $T \leftarrow\!\!\$ \{0,1\}^{\lvert T\rvert}$
$b' \leftarrow\!\!\$\ \mathcal{A}(y)$	return $b' = b$	$\mathrm{L} \leftarrow \mathrm{L} \cup \{M\}$
return $b' = b$		return T

Figure 2.1: Games defining the security of pseudorandom generators (left), pseudorandom functions (right).

2.2.3 Pseudorandom Functions

A PRF is a pair of algorithms $(\mathsf{Setup}, \mathsf{PRF})$, where Setup is a randomized algorithm that on input the security parameter 1^λ generates a key K in some key space $\mathcal{K}$. We will assume that this algorithm simply outputs a random key in $\{0,1\}^\lambda$. Algorithm PRF is deterministic and given K as input and a point $x \in \mathcal{D}$ outputs a value $y \in \mathcal{R}$. We define the advantage of an adversary $\mathcal{A}$ against PRF as

$$\mathsf{Adv}^{\mathrm{prf}}_{\mathcal{A},\mathsf{PRF}}(\lambda) := 2 \cdot \Pr\left[\mathrm{PRF}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda) = 1\right] - 1 \ ,$$

where game $\mathrm{PRF}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda)$ is shown in Figure 2.1 (right). A PRF is secure if the above advantage function in negligible for every PPT adversary $\mathcal{A}$.

Puncturable PRFs. Punctured programming is a novel proof technique proposed by Sahai and Waters in [SW14]. Essentially, it asks one to compute a program in all inputs but one. We will use puncturable PRFs, which allow for a method to puncture the real PRF key K at one point – say M^* – and to obtain a new key K^*.

Definition 2.2 (Puncturable PRFs)**.** *A puncturable pseudorandom function* pPRF *is a tuple of algorithms* $(\mathsf{pPRF.Setup}, \mathsf{pPRF.Eval}, \mathsf{pPRF.Puncture})$ *such that:*

- $K \leftarrow\!\!\$\ \mathsf{Setup}$*: samples K uniformly at random over the key space.*
- $K^* \leftarrow \mathsf{Puncture}(K, M^*)$ *: given K and a point M in the input space, a punctured key K^* is obtained.*
- $Y \leftarrow \mathsf{pPRF.Eval}(K, M)$*: identical to a* PRF*'s' evaluation.*

The correctness requirement states that for $M^* \in \mathcal{M}$, for all $K \in \mathcal{K}$ and for all $M \neq M^* \in \mathcal{M}$, we have that:

$$\mathsf{PRF.Eval}(K, M) = \mathsf{pPRF.Eval}(K^*, M) \ ,$$

where $K^* \leftarrow \mathsf{PRF.Puncture}(K, M^*)$.

We also require the output distribution of the pPRF *to be computationally indistinguishable from the uniform distribution, as depicted in Figure 2.1 (right). Moreover, we require that even in the presence of the punctured key* K^**, a* PPT *adversary cannot distinguish between* $\mathsf{pPRF.Eval}(K, M^*)$ *and* $Y \leftarrow_{\$} \mathcal{R}$.

2.2.4 Symmetric Encryption Schemes

Symmetric encryption algorithms ensure confidentiality of messages under a pre-shared key.

Definition 2.3 (Symmetric Encryption Scheme)**.** *A symmetric encryption scheme* SE *is defined as a triple of* PPT *algorithms* (KGen, Enc, Dec) *such that:*

- *A key generation algorithm* $K \leftarrow_{\$} \mathsf{KGen}(1^\lambda)$ *takes as input the security parameter* λ *in unary and outputs a key* K*;*
- *The encryption algorithm* $C \leftarrow_{\$} \mathsf{Enc}(K, M)$ *gets as input a key* K *and a plaintext* M*, while it outputs a ciphertext* C*;*
- *The deterministic decryption algorithm* $M \leftarrow \mathsf{Dec}(K, C)$ *takes as input a key* K *and a ciphertext* C*, and outputs a plaintext* M.

2.2.5 Public Key Encryption

Public key encryption (PKE) enables encryption in the absence of a pre-shared secret key.

Definition 2.4. *A public key encryption scheme* PKE *consists of a triple of algorithms* (PKE.Setup, PKE.Enc, PKE.Dec) *described as follows:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*: given as input the unary representation of the security parameter* λ*, this algorithm returns a public key* pk *and a secret key* sk.
- $C \leftarrow_{\$} \mathsf{Enc}(\mathsf{pk}, M)$*: the randomized encryption algorithm takes as input* M *and the public key* pk*, producing as output a ciphertext* C.
- $M \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$*: given as input the* C *and the secret key* sk*, the decryption outputs the message* M.

We require any public key encryption algorithm to satisfy correctness for any $M \in \mathcal{M}$*:*

$$\Pr\left[\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{PKE.Enc}(\mathsf{pk}, M)) = M \;\middle|\; (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{PKE.Setup}(1^\lambda) \right] \in 1 - \textsc{Negl}(\lambda) \ .$$

2.2.6 Digital Signature Schemes

Digital signature schemes are standard tools used to publicly validate the authenticity of data.

Definition 2.5 (Digital Signature Scheme)**.** *A digital signature scheme* DS *defined over a message-space* $\mathcal{M}$ *consists of a tuple of four polynomial-time algorithms* (DS.Setup, DS.KGen, DS.Sign, DS.Ver) *such that:*

- $\mathsf{params} \leftarrow_{\$} \mathsf{DS.Setup}(1^\lambda)$*: we assume the existence of a* Setup *algorithm producing a set of public parameters which are implicitly given to all algorithms.*
- $(\mathsf{sk}, \mathsf{vk}) \leftarrow_{\$} \mathsf{DS.KGen}(\mathrm{params})$*: the randomized key generation algorithm takes as input the unary representation of the security parameter* λ *and outputs a pair of secret/verification keys.*
- $\sigma \leftarrow_{\$} \mathsf{DS.Sign}(\mathsf{sk}, M)$*: the (possibly randomized) signing algorithm takes a message* $M \in \mathcal{M}$ *as input and produces a signature* σ *of* M *under the secret key* sk*.*
- $b \leftarrow \mathsf{DS.Ver}(\mathsf{vk}, \sigma, M)$*: the deterministic verification algorithm receives as input a signature* σ *of* M *and checks its validity concerning the verification key* vk *and* M*. It outputs a bit* b*, which represent the verification passes* ($b = 1$) *or not* ($b = 0$)*.*

A digital signature is required to satisfy the following properties:

- *Correctness: for any message* $M \in \mathcal{M}$ *we have that*

$$\Pr\left[\begin{array}{l|l} 1 \leftarrow \mathsf{DS.Ver}(\mathsf{vk}, \sigma, M) & \begin{array}{l}\mathsf{params} \leftarrow_{\$} \mathsf{DS.Setup}(1^\lambda) \wedge \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow_{\$} \mathsf{DS.KGen}(\mathrm{params}) \wedge \\ \sigma \leftarrow_{\$} \mathsf{DS.Sign}(\mathsf{sk}, M)\end{array}\end{array}\right] \in 1 - \textsc{Negl}(\lambda) \ .$$

- *A signature scheme is* EUF*-secure if the advantage of any* PPT *adversary* $\mathcal{A}$ *against the* EUF*-game defined in Figure 2.2 is negligible:*

$$\mathsf{Adv}^{\mathrm{euf}}_{\mathcal{A},\mathsf{DS}}(\lambda) \coloneqq \Pr\left[\mathrm{EUF}^{\mathcal{A}}_{\mathsf{DS}}(\lambda) = 1\right] \in \textsc{Negl}(\lambda) \ .$$

$\underline{\mathrm{EUF}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)}$:	$\underline{\text{Proc. } \mathrm{Sign}_{\mathsf{sk}}(M)}$:
$L \leftarrow \emptyset$	$\sigma \leftarrow_{\$} \mathsf{DS.Sign}(\mathsf{sk}, M)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow_{\$} \mathsf{DS.KGen}(1^\lambda)$	$L \leftarrow L \cup \{M\}$
$(M^*, \sigma^*) \leftarrow_{\$} \mathcal{A}^{\mathrm{Sign}_{\mathsf{sk}}(\cdot)}(1^\lambda, \mathsf{vk})$	return σ
if $M^* \notin L$:	
return $\mathsf{DS.Ver}(\mathsf{vk}, M^*, \sigma^*)$	$\underline{\text{Proc. } \textsc{Ver}_{\mathsf{vk}}(M, \sigma)}$:
return 0	return $\mathsf{DS.Ver}(\mathsf{vk}, M, \sigma)$

Figure 2.2: The existential unforgeability experiment defined for digital signature schemes.

2.2.7 Fully Homomorphic Encryption

Fully homomorphic encryption (FHE) enables computations over encrypted data. The result is another ciphertext, which when decrypted, carries the computations on the original plaintext. It was originally proposed in the work of Rivest, Adleman and Dertouzos [RAD78] and it remained an open problem until the breakthrough work of Gentry [Gen09].

Definition 2.6 (FHE)**.** *A fully homomorphic encryption scheme consists of a tuple* PPT *algorithms* (Setup, Enc, Eval, Dec) *such that:*

- $(\mathsf{hsk}, \mathsf{hpk}) \leftarrow_{\$} \mathsf{Setup}(1^\lambda)$*: a randomized algorithm returning a pair of homomorphic public and secret keys.*

- $C \leftarrow_{\$} \mathsf{Enc}(\mathsf{hpk}, M)$*: the encryption algorithm uses* hpk *to produce a homomorphic ciphertext* C*.*

- $C' \leftarrow \mathsf{Eval}(\mathsf{hpk}, C, f)$*: a function* f *is evaluated over the ciphertext* C*, the resulting being another ciphertext corresponding to* $f(M)$*.*

- $f(M) \leftarrow \mathsf{Dec}(\mathsf{hsk}, C')$*: the decryption is a deterministic procedure that is given the homomorphic secret-key* hsk*, the ciphertext* C' *and reveals* $f(x)$*.*

We say that a FHE *scheme is* ***perfectly correct*** *if for all* $\mathcal{C} : \{0,1\}^k \to \{0,1\}^l$ *of depth* d *and for all* $x \in \{0,1\}^k$ *we have that:*

$$\Pr\left[\begin{array}{l|l} \mathsf{FHE.Dec}(\mathsf{hsk}, \mathsf{FHE.Eval}(\mathcal{C}, & \\ \quad \mathsf{FHE.Enc}(\mathsf{hpk}, x))) = \mathcal{C}(x) & (\mathsf{hpk}, \mathsf{hsk}) \leftarrow_{\$} \mathsf{FHE.Setup}(1^\lambda, 1^d) \end{array}\right] = 1 \ .$$

We also require that (FHE.Setup, FHE.Enc, FHE.Dec) *to constitute a semantic secure public-key encryption scheme.*

2.2.8 Garbling Schemes

Garbling schemes were introduced by Yao in 1986 [Yao86] to solve the famous "Millionaires' Problem". Since then, garbled circuits became a standard building-block for many cryptographic primitives. Their definition follows.

Definition 2.7 (Garbling Scheme)**.** *Let* $\{\mathcal{C}_\lambda\}_\lambda$ *be a family of circuits taking as input* λ *bits. A garbling scheme is a tuple of* PPT *algorithms* (Garble, Enc, Eval) *such that:*

- $(\Gamma, \mathsf{sk}) \leftarrow_{\$} \mathsf{Garble}(1^\lambda, \mathcal{C})$*: takes as input the unary representation of the security parameter and a circuit* $\mathcal{C} \in \{\mathcal{C}_\lambda\}$ *and outputs a garbled circuit* Γ *and a secret key* sk*.*

- $c \leftarrow_{\$} \mathsf{Enc}(\mathsf{sk}, M)$*: is given as input* $M \in \{0,1\}^*$ *and the secret key* sk*, the encoding procedure returns an encoding* c*.*

- $\mathcal{C}(M) \leftarrow \mathsf{Eval}(\Gamma, c)$*: the evaluation procedure receives as inputs a garbled circuit as well as an encoding of* M*, returning* $\mathcal{C}(M)$*.*

We say that a garbling scheme Γ *is* ***correct*** *if for all* $\mathcal{C} : \{0,1\}^k \to \{0,1\}^l$ *and for all* $M \in \{0,1\}^k$ *we have that:*

$$\Pr\left[\mathcal{C}(M) = y \;\middle|\; \begin{array}{l} (\Gamma, \mathsf{sk}) \leftarrow_{\$} \mathsf{GS.Garble}(1^\lambda, \mathcal{C}) \wedge \\ y \leftarrow \mathsf{GS.Eval}(\Gamma, \mathsf{GS.Enc}(\mathsf{sk}, M)) \end{array}\right] = 1 \ .$$

Yao's Garbling Scheme. An interesting type of garbled schemes is represented by the original proposal of Yao, which considers a family of circuits of n input wires and outputting one single bit. In his proposal, a circuit's secret key can be viewed as two labels (L_i^0, L_i^1) for each input wire, where $i \in [n]$. The evaluation of the circuit at point x corresponds to an evaluation of $\mathsf{Eval}(\Gamma, (L_1^{x_1}, \ldots, L_n^{x_n}))$, where x_i is the i^{th} bit of x, — thus the encoding $c = (L_1^{x_1}, \ldots, L_n^{x_n})$.

2.2.9 Attribute-Based Encryption

Attribute-based encryption (ABE) (in the *key-policy* setting) is a particular case of (public-key) functional encryption (Definition 5.3). A (functional) key is generated for a predicate γ, while a ciphertext is the encryption of a set of attributes α. Thus, the owner of the (functional) key can recover the secret message encrypted with some attributes α if $\gamma(\alpha) = 1$, or nothing otherwise.

Definition 2.8 (ABE [GPSW06])**.** *A key-policy attribute-based encryption scheme is a tuple of* PPT *algorithms such that:*

- $(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*: takes as input the unary representation of the security parameter λ and outputs the master public key* mpk *and a master secret key* msk*.*
- $\mathsf{sk}_\gamma \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, \gamma)$*: given the master secret key and a policy γ, the (potentially randomized) key-derivation outputs a corresponding* sk_γ*.*
- $C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, \alpha, M)$*: the randomized encryption procedure encrypts the plaintext M with respect to some attribute set α.*
- $\mathsf{Dec}(\mathsf{sk}_\gamma, C)$*: decrypts the ciphertext C using the key* sk_γ *and obtains M if $\gamma(\alpha) = 1$ or a special symbol $\perp$, in case the decryption procedure fails (i.e. $\gamma(\alpha) = 0$).*

In their work, Goldwasser et al. [GKP+13] extend the notion of ABE to a new primitive, dubbed as two-outcome attribute-based encryption (ABE_2), which distinguishes itself through the way encryption and decryption proceed:

Definition 2.9 (Two-Outcome Attribute-Based Encryption)**.** *A Two-Outcome Attribute-Based Encryption scheme* ABE_2 *is identical to the key-policy* ABE *scheme up to:*

- $C \leftarrow\!\!\$\ \mathsf{ABE}_2.\mathsf{Enc}(\mathsf{mpk}, \alpha, M_0, M_1)$*: the randomized encryption procedure encrypts two plaintexts with respect to some attribute set α.*
- $\mathsf{ABE}_2.\mathsf{Dec}(\mathsf{sk}_\gamma, C)$*: decrypts the ciphertext C using the key* sk_γ *and obtains M_0 if $\gamma(\alpha) = 0$ or M_1 if $\gamma(\alpha) = 1$.*

2.2.10 Functional Encryption

Functional encryption[BSW11; ONe10b] is one of the most general encryption paradigms, as it encompasses attribute-based, identity-based or public-key encryption as particular cases. Concretely, an FE scheme allows for surgical access over encrypted data, controlling the leak the adversary sees: ciphertexts correspond to messages M, keys are derived for functions f, while adversaries are able to learn $f(M)$ and (ideally) nothing more. Different paradigms were considered to the date: (1) public vs private schemes, (2) single or multi-input ones, (3) function-revealing or function-hiding constructions. Several versions of FE are known to imply iO for general circuits (such as compact FE [AJ15; BV15] or multi-input FE in the public-key setting). In our work, we present construction of UBK-secure pseudorandom permutations, mostly using the power of FE to hide the keys and to evaluate the circuits (embedded in the functional keys).

Definition 2.10 (Functional Encryption Scheme - Public-Key Setting)**.** *A functional encryption scheme* FE *in the public-key setting consists of a tuple of* PPT *algorithms* (PPGen, Setup, KGen, Enc, Dec) *such that:*

- $\mathsf{params} \leftarrow_{\$} \mathsf{FE.PPGen}(1^\lambda)$*: we assume the existence of a* PPGen *algorithm producing a set of public parameters which are implicitly given to all algorithms. When omitted from description, we assume* $\mathsf{params} \leftarrow 1^\lambda$*.*
- $(\mathsf{msk}, \mathsf{mpk}) \leftarrow_{\$} \mathsf{FE.Setup}(\mathsf{params})$ *: takes as input the public parameters and outputs a pair of master secret/public keys.*
- $\mathsf{sk}_f \leftarrow_{\$} \mathsf{FE.KGen}(\mathsf{msk}, f)$*: given the master secret key and a function* f*, the (randomized) key-derivation procedure outputs a corresponding functional key* sk_f*.*
- $C \leftarrow_{\$} \mathsf{FE.Enc}(\mathsf{mpk}, M)$*: the randomized encryption procedure encrypts the plaintext* M *with respect to* mpk*.*
- $\mathsf{FE.Dec}(C, \mathsf{sk}_f)$*: decrypts the ciphertext* C *using the functional key* sk_f *in order to learn a valid message* $f(M)$ *or a special symbol* $\perp$*, in case the decryption procedure fails.*

A functional encryption scheme is s-IND-FE-CPA*-secure if the advantage of any* PPT *adversary* $\mathcal{A}$ *against the* IND-FE-CPA*-game defined in Figure 2.4 is negligible:*

$$\mathsf{Adv}^{\text{s-ind-fe-cpa}}_{\mathcal{A},\mathsf{FE}}(\lambda) \coloneqq 2 \cdot \Pr\left[\text{s-IND-FE-CPA}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) = 1\right] - 1 \in \text{Negl}(\lambda) \ .$$

Similarly we say that it is adaptive IND-FE-CPA*-secure if*

$$\mathsf{Adv}^{\text{ind-fe-cpa}}_{\mathcal{A},\mathsf{FE}}(\lambda) \coloneqq 2 \cdot \Pr\left[\text{IND-FE-CPA}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) = 1\right] - 1 \in \text{Negl}(\lambda) \ .$$

FULL-SIM-FE ***Security.*** *For the particular case of single input functional encryption schemes, we recall simulation-based security as introduced in [GKP+13]. We say a public-key functional encryption scheme* FE *is semantically secure if there exists a stateful* PPT *simulator* $\mathcal{S}$ *such that for any* PPT *adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{\text{full-sim-fe}}_{\mathcal{A},\mathsf{FE}}(\lambda) \coloneqq 2 \cdot \Pr[\mathsf{FULL\text{-}SIM\text{-}FE}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) = 1] - 1 \in \text{Negl}(\lambda) \ ,$$

where the FULL-SIM-FE *experiment is described in Figure 2.3.*

Private-Key Setting. FE was initially defined in the public-key setting. However, it turns out the case for private-key FE is also interesting: in conjunction with PKE, it implies iO for circuits with inputs of poly-logarithmic length [KS17]. We do not discuss the advantages or limitations of these two models. For completeness, we define the scheme below:

Definition 2.11 (Functional Encryption Scheme — Private-Key Setting)**.** *A functional encryption scheme* FE *is a tuple of* PPT *algorithms* (FE.Setup, FE.KGen, FE.Enc, FE.Dec) *such that:*

- $\mathsf{msk} \leftarrow_{\$} \mathsf{FE.Setup}(1^\lambda)$ *: takes as input the unary representation of the security parameters and outputs* msk*.*
- $\mathsf{sk}_f \leftarrow_{\$} \mathsf{FE.KGen}(\mathsf{msk}, f)$*: given the master secret key and a function* f*, the (randomized) key-derivation procedure outputs a corresponding* sk_f*.*

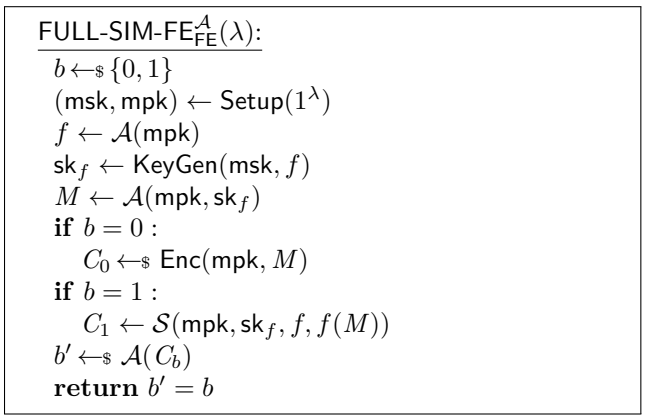

Figure 2.3: The FULL-SIM-FE security for public-key functional encryption schemes, as defined in [GKP+13].

- $C \leftarrow\!\!\$ \mathsf{FE.Enc}(\mathsf{msk}, M)$*: the randomized encryption procedure encrypts the plaintext* M *with respect to* msk*.*
- $\mathsf{FE.Dec}(C, \mathsf{sk}_f)$*: decrypts the ciphertext* C *using the functional key* sk_f *in order to learn a valid message* $f(M)$ *or a special symbol* $\perp$*, in case the decryption procedure fails.*

A functional encryption scheme is IND-FE-CPA*-secure if the advantage of any* PPT *adversary* $\mathcal{A}$ *against the* IND-FE-CPA*-game defined in Figure 2.4 is negligible:*

$$\mathsf{Adv}^{\text{ind-fe-cpa}}_{\mathcal{A},\mathsf{FE}}(\lambda) \coloneqq 2 \cdot \Pr\left[\text{IND-FE-CPA}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) = 1\right] - 1 \in \text{NEGL}(\lambda) \ .$$

2.2.11 Indistinguishability Obfuscation

We use the formal indistinguishability definition for an obfuscator of a class of circuits [Lin17].

Definition 2.12 (Indistinguishability Obfuscation (iO) for a circuit class)**.** *A uniform* PPT *machine* iO *is an indistinguishability obfuscator for a class of circuits* $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *if the following conditions are satisfied:*

- ***Correctness****:*
$$\Pr\left[\forall x \in \mathcal{D}, C(x) = \overline{\mathbf{C}}(x) | \overline{\mathbf{C}} \leftarrow\!\!\$ \ \mathsf{iO}(C)\right] = 1 \ .$$
- ***Indistinguishability****:*
$$\left| \Pr\left[b = b' \;\middle|\; \begin{array}{l} \forall C_1, C_2 \in \{C\}_\lambda \wedge \forall x \in \mathcal{D} : C_1(x) = C_2(x) \\ \quad \wedge\ b \leftarrow\!\!\$ \{0,1\} \ \wedge\ \overline{\mathbf{C}} \leftarrow\!\!\$ \ \mathsf{iO}(C_b) \\ \quad \wedge\ b' \leftarrow\!\!\$ \ \mathcal{A}(1^\lambda, \overline{\mathbf{C}}, C_0, C_1) \end{array} \right] - \frac{1}{2} \right| \in \text{NEGL}(\lambda).$$

where $\mathcal{D}$ *is the input domain of the circuits* C*.*

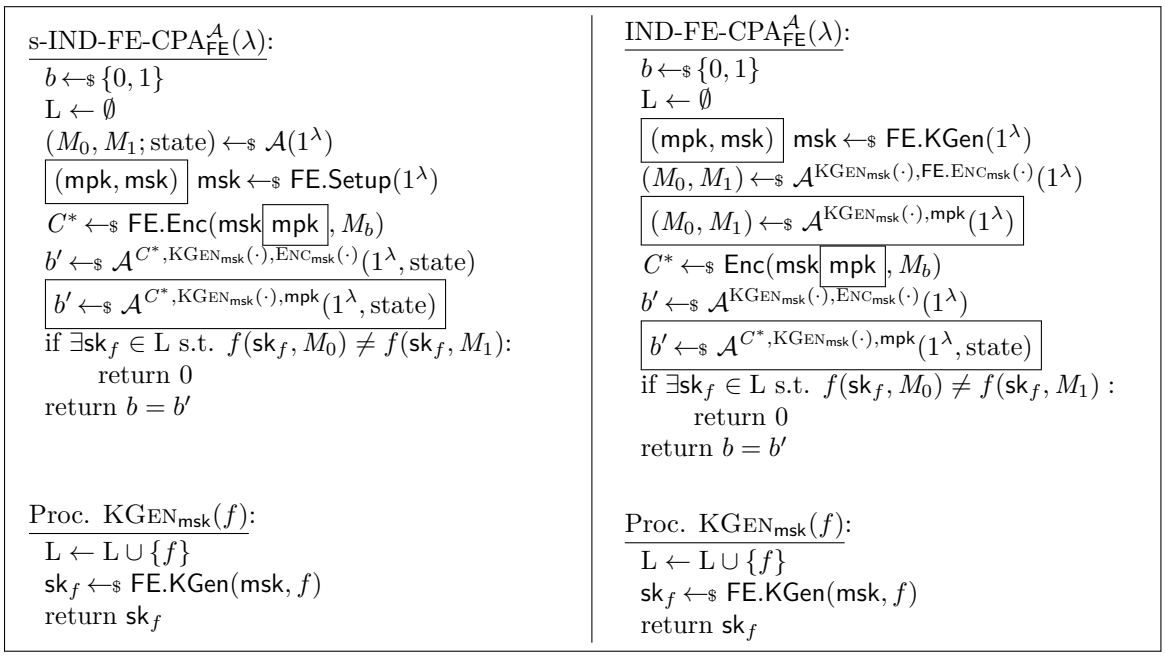

Figure 2.4: The *selective* and *adaptive* indistinguishability experiments defined for a functional encryption scheme. The difference between the private-key and the public settings are marked in boxed lines of codes, corresponding to the latter notion.

2.3 Computational Hardness Assumptions

Below, we introduce the main computational hardness assumptions to be used herein.

2.3.1 Discrete-Log Related Assumptions

Definition 2.13 (Multiple-DDH problem [BCP02])**.** *Let $n \geq 2$, $g \in \mathbb{G}$ be a generator for the cyclic group $\mathbb{G}$ of prime order p, and $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ be elements sampled uniformly at random. The following advantage of any* PPT *adversary $\mathcal{A}$ is negligible:*

$$\mathsf{Adv}^{n\text{-}\mathsf{DDH}}_{\mathcal{A}}(\lambda) \coloneqq \Pr\left[1 \leftarrow_\$ \mathcal{A}\left(1^\lambda, (g^{x_1}, \ldots, g^{x_n}, \{g^{x_i x_j}\}_{1 \leq i < j \leq n})\right)\right] - \Pr\left[1 \leftarrow_\$ \mathcal{A}\left(1^\lambda, (g_1, \ldots, g_n, \{g_{i,j}\}_{1 \leq i < j \leq n})\right)\right] \in \mathrm{NEGL}(\lambda)\ .$$

2.3.2 Bilinear Maps

Bilinear maps found numerous applications in public-key cryptography. We define them below.

Definition 2.14. *Let $(\mathbb{G}_1, \cdot), (\mathbb{G}_2, \cdot), (\mathbb{G}_T, \cdot)$ be cyclic groups of prime order p. Let g_1, g_2 be the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. We call $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ a bilinear map if e is efficiently computable and:*

- *Given any $(a, b) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, we have that $e(g_1^a, g_2^b) = e(g_1, g_2)^{a \cdot b}$.*
- *Non-degeneracy: $e(g_1, g_2) \neq 1$.*

Bilinear maps found applications in the realization of signature schemes. We review an assumptions used in the [BB08] signature scheme based on cryptographic pairings:

Definition 2.15 (q-Strong Diffie-Hellman problem [BB08])**.** *Let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic groups of prime order p with generators g_1, g_2 and let $x \leftarrow\!\!\$\ Z_p$. The following advantage of any* PPT *adversary $\mathcal{A}$ is negligible:*

$$\mathsf{Adv}_{\mathcal{A}}^{q\text{-}\mathsf{SDH}}(\lambda) \coloneqq \Pr\left[(c, g_1^{1/(x+c)}) \leftarrow\!\!\$\ \mathcal{A}\left(1^{\lambda}, (g_1, g_2, g_2^{x}, \ldots, g^{x^q})\right) \wedge c \in \mathbb{Z}_p^*\right] \in \textsc{Negl}(\lambda)\ .$$

2.3.3 Collision-Resistant Hash Functions

We also use regular collision-resistant hash functions as described by Canetti, Micciancio and Reingold in [CMR98]. These are hash functions exhibiting an extra property of regularity and can be constructed from claw-free permutation pairs.

Definition 2.16 (Regular Collision-Resistant Hash Function [CMR98])**.** *A function $h : \mathcal{D} \to \mathcal{R}$ is regular if the random variable $h(x)$ defined by a uniformly distributed $x \in \mathcal{D}$, is uniform over $\mathcal{R}$: for all $y \in \mathcal{R}$, $|h^{-1}(y)| = |\mathcal{D}|/|\mathcal{R}|$ holds. A regular collision-resistant hash function h is a collision-resistant hash meeting the regularity condition.*

2.3.4 Learning With Errors and Related Assumptions

The Learning With Error (LWE) search problem [Reg05] asks for the secret vector $\mathbf{s}$ over $\mathbb{Z}_q^n$ given a set of noisy vectors of the form $\mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e}$, where $\mathbf{A}$ denotes a randomly sampled matrix over $\mathbb{Z}_q^{n \times m}$, while $\mathbf{e}$ is a small error term sampled from an appropriate distribution χ. Roughly speaking, the decision version of the problem asks to distinguish between the distribution of the LWE problem as opposed to the uniform distribution.

Definition 2.17 (Learning with Errors)**.** *For an integer $q = q(\lambda) \geq 2$ and an error distribution $\chi = \chi(\lambda)$ over $\mathbb{Z}_q$, the decision learning with errors problems is to distinguish between the following pairs of distributions:*

$$\{(\mathbf{A}, \mathbf{A}^\top \cdot \mathbf{s} + \mathbf{e})\} \quad \textit{and} \quad \{(\mathbf{A}, \mathbf{u})\}$$

where $\mathbf{A} \leftarrow\!\!\$\ \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow\!\!\$\ \mathbb{Z}_q^n, \mathbf{e} \leftarrow_\chi \mathbb{Z}_q^m, \mathbf{u} \leftarrow\!\!\$\ \mathbb{Z}_q^m$.

Later, Regev *et al.* [LPR10] proposed a version over quotient rings: let $R = \mathbb{Z}[X]/(X^n + 1)$ for n a power of 2, while $R_q \coloneqq R/qR$ for a safe prime q satisfying $q = 1 \bmod 2n$. An adversary is required to distinguish between the following distributions: $\{(a, a \cdot s + e)\}$ and $\{(a, u)\}$, for a, s, u elements sampled independently and uniformly at random over R_q and e a small error term.

Chapter 3

Robust Encryption

Robustness for public-key encryption guarantees that a ciphertext cannot be decrypted under two (or more) distinct keys. In this chapter, we generalize the definitions of robust PKE to the functional encryption setting and provide simple, generic transforms that turn FE schemes into robust ones. Along the way, we rely on robustness for pseudorandom functions, a security notion we put forward.

3.1 Overview of Key-Robustness

Cryptographic primitives, such as encryption and signature schemes, provide security guarantees under the condition, often left implicit, that they are "used correctly". Fatal examples of cryptographic misuse abound, from weak key generation to nonce reuse. This reliance on operational security has attracted attackers, who can, for instance, impose faulty or backdoored random number generators to erode cryptographic protections. At the same time, the social usage of technology leans towards a more open environment than the one in which traditional primitives were designed: keys are generated by one party, shared with another, certified by third... These two observations raise new interesting questions, which have only recently been addressed in the cryptographic literature. For instance, if Alice generates keys that she is using, but doesn't share, can an adversary (observing Alice or influencing her in some way) nevertheless generate a *different* set of keys, which would allow decryption (maybe only partial)? Intuitively this should not be the case, but it was not until the seminal work of Abdalla, Bellare and Neven [ABN10; ABN18], that this situation was formally analysed. They introduced the notion of robustness, which ensures that a ciphertext cannot be decrypted under multiple keys.

Is robustness desirable? Imagine a scenario where users within a network exchange messages by broadcasting them, and further encrypt them with the public key of the recipient to ensure confidentiality. If this is the case, we usually assume that there is only one receiver, by arguing that no other members apart from the intended recipient can decrypt the ciphertext and obtain a valid (non-$\perp$) plaintext. However, if the adversary can somehow tamper with the key generation process, she may "craft" keys that behave unexpectedly for some messages or design alternative keys that give at least some information on some of the messages.

Farshim et al. [FLPQ13] refined the original definition of robustness, by covering the cases where the keys are adversarially generated, under a master notion called "complete robustness". Mohassel addressed the question in the context of key-encapsulation mechanisms [Moh10]. More recently, Farshim et al. put forward robustness for symmetric primitives [FOR17], motivated by the security of oblivious transfer protocols [CO15] or message authentication codes. Further extensions of their security notions found applications in the original password-authenticated key-exchange protocols described by Jarecki et al. [JKX18] or (fast) message-franking schemes [GLR17]. We review the original motivating example and point out that the above line of works leaves several open questions on the semantic of this security definition in the context of functional encryption [BSW11; ONe10a] or digital signature schemes [GMR84; BGI14] (Chapter 4). We provide a second motivating example that suggests robustness is an excellent companion of any FE scheme.

Example 1 – Anonymous Communication. Many practical symmetric encryption schemes have ciphertexts that look random, which in particular implies a form of *key anonymity*: when given two ciphertexts – C_0, C_1 – it is hard to tell whether or not they were generated using the same (unknown) secret key. Imagine a protocol with one sender and several receivers, where each receiver shares a key k_i with the sender. Anonymity guarantees that if the sender broadcasts a ciphertext constructed using k_i, then a different receiver j should only learn that $i \neq j$ and nothing else. At the same time, such protocols often intuitively assume that at most one of the receivers is believed to be the intended receiver, i.e., decryption will fail *for all but one* of the users. However, this is not covered by standard security definitions. Similarly, many public-key encryption constructions produce ciphertexts

indistinguishable from random, meaning the previous scenario may repeat if PKE schemes would be used. More generally, whenever user anonymity is a security goal, it is likely that some form of robustness is also needed in order to avoid undesired behaviour [ABN10].

Example 2 – Robustness for Inner-Product Encryption: The previous example involving standard, symmetric/public-key encrypted and exchanged messages between multiple parties can be further generalized. Consider a simple use case of a functional encryption scheme for the "inner product" function (IP FE) [ABDP15]. From a technical perspective, suppose the ciphertext is generated by encrypting a plaintext M as $C \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, M; R)$. If msk is somehow corrupted[1] to msk', then is it possible that performing decryption under sk'_y reveals a different plaintext $M' \neq M$? Intuitively, if the functional encryption scheme meets robustness, we expect that no ciphertext can decrypt under functional keys issued under *different* master secret keys.

As a concrete scenario, consider a Computer Science (CS) department's registry, which holds the marks obtained by each student in the Crypto course. The final grade is being computed as a weighted average of the stored marks (i.e., homework counts 30%, midterm 20% and final 50%). *A priori* established confidentiality rules ask that a clerk should not have access to the marks, but still, it must be possible to compute the final grade. Therefore, considering the set of marks as the vector $\vec{x}$ and the weights as $\vec{y}$, one can use an IP FE scheme, to obtain the final grade, its formula mapping to $\vec{x}^\top \cdot \vec{y}$. In order to achieve this, for each course: (1) the course leader encrypts the marks; (2) later, the clerk obtains a new key sk_y (depending on the established course weights), and uses it to obtain the final average. A failure to guarantee robustness could result in decryption to succeed, but the final average being incorrect (and possibly under the control of an adversary). To illustrate this, consider the (bounded-norm) IP FE scheme instantiated from ElGamal encryption and introduced in [ABDP15]: encrypting a plaintext under $\mathsf{mpk} = (g^{s_1}, \ldots, g^{s_n})$ — where $\mathsf{msk} = \vec{s} \leftarrow (s_1, \ldots, s_n)$ — is done as follows: $C \leftarrow (g^{-r}, g^{r \cdot s_1 + x_1}, \ldots, g^{r \cdot s_n + x_n})$, for r sampled uniformly at random in $\mathbb{Z}_p$. If an attacker wishes to obtain the same C, then r remains the same, but it can use different $\vec{s'}$ and $\vec{x'}$, implicitly changing the value of msk. As expected, even if $\mathsf{FE.KGen}$ is correct, and the queried key is indeed issued for the vector $\vec{y}$, the final decrypted result corresponds to $\vec{x'}^\top \cdot \vec{y}$ rather than to $\vec{x}^\top \cdot \vec{y}$.

3.1.1 Previous Work on Robustness

In this chapter, we interact with a rather small subset of the possible, relevant definitions encapsulating the intuition behind robustness. In a sense, we attempt to capture realistic attacks. However, from a theoretical perspective, a more general discussion would consider a comprehensive set of definitions for robustness.

A first option is to study the guarantees obtained under honestly generated keys. We expect that for many encryption primitives, the standard correctness and indistinguishability security notions are enough to formally prove that a ciphertext cannot be decrypted under multiple keys. The original work of [ABN10] introduces this notion for public-key encryption under the name of strong-robustness (SROB).

The next natural step is to consider settings with increased adversarial power; this can be done by considering robustness notions where *adversarial key-generation* is possible. At

[1] There are several scenarios leading to such corruption, including memory corruption.

this point, we distinguish two paths: (1) the adversary can interact with the key generation process and set/inspect partial bits in a key; (2) the adversary can generate the key in its entirety. An analysis of the former case would cover a more realistic class of attacks, incorporating the ones presented by Heninger and Shacham [HS09], while the latter favours simplicity.

Furthermore, one can reconsider the original definition of robustness: *a ciphertext cannot be decrypted under multiple keys* into a setting in which, under adversarial key generation, the adversary is asked to find colliding ciphertexts by issuing encryption keys, randomness terms and plaintexts. [FLPQ13] elaborates more on such definitions for public-key encryption, by formalizing the notions of:

- Keyless Robustness (KROB): the adversary outputs two public-keys, two messages and two randomness terms, winning if it obtains two colliding ciphertexts. Thus, the security experiment uses encryption queries only.
- Full Robustness (FROB): an adversary outputs a ciphertext C and two secret-keys $\mathsf{sk}_1, \mathsf{sk}_2$, winning if C decrypts to two valid messages under the two different secret-keys. This experiment uses only decryption queries.
- Mixed Robustness (XROB): is a variation between the two, essentially requiring one encryption and one decryption evaluation.
- Complete Robustness (CROB): is the "union" of the previous three security notions, by considering adversaries that can mount FROB, XROB and KROB attacks.

Finally, the existing robustness notions can be extended to more general primitives, such as identity-based encryption [Sha84; BF01], attribute-based [SW05] encryption or functional encryption [BSW11; ONe10b]. As an informal rule, the more advanced the primitive, the more convoluted the definition. Formalizing such definitions and providing transforms for achieving robustness constitutes the main contribution we give throughout this chapter.

3.1.2 Chapter Organization

This chapter is based on the works published by the author in [FOR17] and [GNR19]. Its structure follows:

- In Section 3.2 we review the existing notions of robustness for public-key encryption, focusing on strong and complete robustness (abbreviated SROB and CROB). We provide similar definitions in the functional setting, in a multi-authority context and discuss why similar definitions cannot be achieved in a single authority context. These notions are designed to enfold the maximal strength of an adversary by allowing to generate the keys and the random coins used for encryption and key-derivation while maintaining syntactical simplicity.
- A natural question is whether existing schemes already possess a form of robustness: we show that while SROB is implied by CROB, there exist FE schemes that are not CROB-secure.
- In Section 3.3, we look into generically achieving robustness. For the case of functional encryption considered in the public-key setting, in addition to a commitment scheme

we make use of an IND-CPA public-key encryption scheme. Turning to the case of a private-key FE scheme, our technique relies on right-injective PRGs and robust PRFs. To a certain extent these transformations are "natural", but we have to ensure and prove that they work as intended.

- Finally, we show how to construct (computationally) robust and collision-resistant PRFs assuming the existence of right injective PRGs (Section 3.4). We think such notions are of independent interest and may find applications in other fields.

3.2 Definitions and Relations

In this section, we give an overview of the generic transform proposed in the original work of Abdalla, Bellare and Neven [ABN10] as well as its refinement by Farshim, Larraia, Quaglia and Patterson [FLPQ13]. Then, we put forward robustness for functional encryption, focusing on a multi-authority context.

3.2.1 Robustness for Public-Key Encryption

The original motivation for introducing robustness resided in Sako's *auction* protocol [Sak00], working as follows: in a preliminary phase, the auctioneer prepares a set of potential bid values $\mathcal{V} = \{v_1, \ldots, v_n\}$. Next, he/she generates n public/secret key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$. During the auction, whenever one participant wants to auction off for some amount i, he/she will write a message (say his/her name plus some identifier) and encrypt it under pk_i. Once the auction ends, the auctioneer decrypts the ciphertext(s) that were received, first under sk_n, (and checks for the winner(s)), then under sk_{n-1} (and checks for the winner(s)) and so on. Abdalla *et al.* observe the public-key scheme used must ensure some kind of robustness guarantees, pleading for the notion of *strong* robustness. While the intuition in [ABN10] is correct, Farshim, Larraia, Quaglia and Paterson [FLPQ13] prove that strong robustness is still insufficient for Sako's protocol, as the keys could be generated maliciously. Therefore, they capture such a scenario by introducing complete robustness.

Strong robustness. Strong robustness captures the ability of ciphertexts to be decrypted under multiple, but honestly generated keys. The game depicted in Figure 3.1 gives an adversary $\mathcal{A}$ the ability to interact with encryption and decryption procedures, winning if $\mathcal{A}$ finds a "problematic" ciphertext C where $\mathsf{PKE.Dec}(\mathsf{sk}_1, C) \neq \perp \wedge \mathsf{PKE.Dec}(\mathsf{sk}_2, C) \neq \perp$.

3.2.1.1 The ABN Transform - Intuition

Abdalla, Bellare and Neven propose a generic transform – the "ABN" transform – aimed at transforming any weak-robust PKE encryption scheme into a strong robust one by using a commitment scheme. By *weak robust* we mean that under honest key generation it is hard for an adversary to come with a message and two public keys $(M, \mathsf{pk}_1, \mathsf{pk}_2)$ such that encrypting M under pk_1 produces a ciphertext C decryptable under a second decryption oracle (decryption is done w.r.t. $\mathsf{sk}_2, \mathsf{pk}_2$). We defer a description of weak-robustness, as it plays no role in our work. Informally, the ABN transform uses a common reference string and proceeds as follows: a commitment scheme CS is instantiated, and the crs is included in the public parameters. Then, the transform commits to the public-key pk, and encrypts the resulting decommitment dec, together with the plaintext. Thus, $C \leftarrow\!\!\$ (\mathsf{com}, \overline{\mathsf{PKE}}.\mathsf{Enc}(\mathsf{pk}, M||\mathsf{dec}))$.

$\underline{\text{SROB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$\mathsf{params} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$C \leftarrow\!\!\$\ \mathcal{A}^{\text{DEC}_{\mathsf{sk}_1}(\cdot),\text{DEC}_{\mathsf{sk}_2}(\cdot)}(\mathsf{params}, \mathsf{pk}_1, \mathsf{pk}_2)$
if $\mathsf{pk}_0 = \mathsf{pk}_1$:
    return 0
$M_1 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, C, \mathsf{pk}_1, \mathsf{params})$
$M_2 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, C, \mathsf{pk}_2, \mathsf{params})$
if $M_1 \neq \perp \wedge M_2 \neq \perp$:
    return 1
return 0

$\underline{\text{KROB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$\mathsf{params} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(\mathsf{pk}_1, M_1, R_1, \mathsf{pk}_2, M_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
if $\mathsf{pk}_0 = \mathsf{pk}_1$:
    return 0
$C_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_1, M_1, R_1, \mathsf{params})$
$C_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, M_2, R_2, \mathsf{params})$
if $C_1 = C_2 \wedge C_1 \neq \perp$:
    return 1
return 0

Figure 3.1: The strong robustness game (left), as defined in [ABN10]. Keyless robustness (KROB), as defined in [FLPQ13].

$\underline{\text{FROB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$\mathsf{params} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(C, \mathsf{pk}_1, \mathsf{sk}_1, \mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
if $\mathsf{pk}_0 = \mathsf{pk}_1$:
    return 0
$M_1 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, C, \mathsf{pk}_1, \mathsf{params})$
$M_2 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_2, C, \mathsf{pk}_2, \mathsf{params})$
if $M_1 \neq \perp \wedge M_2 \neq \perp$:
    return 1
return 0

$\underline{\text{XROB}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$:
$\mathsf{params} \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(C_1, \mathsf{pk}_1, \mathsf{sk}_1, \mathsf{pk}_2, M_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
if $\mathsf{pk}_0 = \mathsf{pk}_1$:
    return 0
$M_1 \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_1, C, \mathsf{pk}_1, \mathsf{params})$
$C_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2, M_2, R_2, \mathsf{params})$
if $M_1 \neq \perp \wedge M_2 \neq \perp \wedge C_1 = C_2$:
    return 1
return 0

Figure 3.2: The enhanced security notion capturing adversarial key generation.

When decrypting, one recovers dec, then checks that $\mathsf{Ver}(\mathsf{crs}, \mathsf{com}, \mathsf{dec}, \mathsf{pk}) = 1$, down to the binding property of the commitment scheme.

3.2.1.2 The Refined ABN Transform for Adversarial Keys

As observed by Farshim et al. in [FLPQ13], the notion of strong robustness introduced in [ABN10] does not suffice to the original goal of protecting losers' bids in Sako's auction protocol, but it still serves as the workhorse of their transform for the notion of complete robustness.

The need for stronger definitions. Thus, [FLPQ13] proposed the three cases of XROB, KROB and FROB notions under adversarial key generation, which are provided in Figures 3.1 and 3.2. Finally, a master notion – dubbed complete robustness (abbreviated CROB) – unionises over the three.

FLPQ transform. As stated previously, SROB is not sufficient for the original application of auction protocols. The transform in [FLPQ13] is akin to the one in [ABN10] up to the significant difference that it handles maliciously generated keys by using a PKE scheme that supports labels (e.g. Cramer-Shoup [CS98]).

3.2.2 Generalizing Robustness

As discussed in the motivational part of Section 3.1, robustness should be considered as a security notion achieved by a *functional encryption* scheme. In what follows, we define it for the public/private key settings.

Speaking roughly about robustness as the property of a ciphertext of not being decryptable under multiple keys, then, when it comes to decryption, an FE scheme trivially does not exhibit this property. The reason resides in the broken symmetry to the way decryption works in symmetric/public-key schemes. Through its purpose, a functional ciphertext can be decrypted under multiple keys [BSW11; ONe10a]. In this respect, an adversary holding multiple functional keys (which is not a restriction by itself) will be able to decrypt under multiple keys. Therefore, defining robustness in terms of decryption itself is fallacious. We stress about the existence of essentially two major paths one can explore – multi-authority or single-authority.

3.2.2.1 Multi-Authority Setting

A first path is placed in a multi-authority context – that is, we assume there exist multiple pairs $(\mathsf{msk}, \mathsf{mpk})$. Aiming for a correct definition, one property that should be guaranteed is that a ciphertext should not be decryptable under (at least) *two* functional keys issued via *different* master secret keys. Stated differently, if msk_1 produces sk_{f_1} and msk_2 ($\neq \mathsf{msk}_1$) produces sk_{f_2} for two functionalities f_1, f_2, we do not want C (say encrypted under mpk_1) to be decrypted under sk_{f_2} (it already decrypts under sk_{f_1} with high probability due to the correctness of the scheme). We propose two *main* flavours of robustness, corresponding to the public and private key settings. Depending on the case, the adversary has oracle access to the (encryption, if in a private key case), key-derivation and decryption oracles. The security experiments are depicted in Figure 3.3. The difference between the two paradigms may seem minor (for our purpose), but in fact, having a *public* master key confers a significant advantage when it comes to deriving a generic transform for achieving complete robustness, as detailed in Section 3.3. In what follows, we explore this path, since it naturally maps to our motivational examples.

Malformed Public Keys. In our experiments, we consider only well-formed master public and secret keys. We believe that it is relatively easy to check if a key is malformed. Thus we reject the style of the artificial separations introduced in [FLPQ13].

SROB and FEROB. As stated in the algorithmic description of the security experiment, an adversary against the strongest notion of FEROB attempts to find colliding ciphertexts, which decrypt under two msk-separated keys $\mathsf{sk}_{f_1}, \mathsf{sk}_{f_2}$.

Definition 3.1 (SROB and FEROB Security for FE)**.** *Let* FE *be a functional encryption scheme. We say* FE *achieves functional robustness if the advantage of any* PPT *adversary* $\mathcal{A}$ *against the* FEROB *game defined in Figure 3.3 (bottom) is negligible:*

$$\mathsf{Adv}^{\mathrm{ferob}}_{\mathcal{A},\mathsf{Pub}/\mathsf{PrvFE}}(\lambda) \coloneqq \Pr\left[\mathrm{FEROB}^{\mathcal{A}}_{\mathsf{Pub}/\mathsf{PrvFE}}(\lambda) = 1\right]$$

SROB*-security is defined similarly, the* $\mathrm{SROB}^{\mathcal{A}}_{\mathsf{Pub}/\mathsf{PrvFE}}(\lambda)$ *game being depicted in Figure 3.3 (top).*

$\underline{\text{SROB}^{\mathcal{A}}_{\mathsf{PubFE}}(\lambda)}$:
$\mathrm{L}_1 \leftarrow \emptyset$
$\mathrm{L}_2 \leftarrow \emptyset$
$(\mathsf{mpk}_1, \mathsf{msk}_1) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$(\mathsf{mpk}_2, \mathsf{msk}_2) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$(C, \mathsf{sk}_{f_1}, \mathsf{sk}_{f_2}) \leftarrow\!\!\$$
$\mathcal{A}\left(\begin{array}{l} \\ \\ \\ \mathrm{KG}_{\mathrm{EN}\mathsf{msk}_1}(\cdot), \\ \mathrm{KG}_{\mathrm{EN}\mathsf{msk}_2}(\cdot) \end{array}\right)(\mathsf{mpk}_1, \mathsf{mpk}_2)$
if $\mathsf{sk}_{f_1} \in \mathrm{L}_2 \vee \mathsf{sk}_{f_2} \in \mathrm{L}_1$:
return 0
if $\mathsf{Dec}(C, \mathsf{sk}_{f_1}) \neq \perp \wedge$
$\mathsf{Dec}(C, \mathsf{sk}_{f_2}) \neq \perp$:
return 1
return 0

$\underline{\mathrm{KG}_{\mathrm{EN}\mathsf{msk}_i}(f)}$:
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, f)$
$\mathrm{L}_i \leftarrow \mathrm{L}_i \cup \{(\mathsf{sk}_f, f)\}$
return sk_f

$\underline{\mathrm{E}_{\mathrm{NC}}\mathsf{mpk}_i(M)}$:
$C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_i, M)$
return C

$\underline{\text{SROB}^{\mathcal{A}}_{\mathsf{PrvFE}}(\lambda)}$:
$\mathrm{L}_1 \leftarrow \emptyset$
$\mathrm{L}_2 \leftarrow \emptyset$
$\mathsf{msk}_1 \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$\mathsf{msk}_2 \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$(C, \mathsf{sk}_{f_1}, \mathsf{sk}_{f_2}) \leftarrow\!\!\$$
$\mathcal{A}\left(\begin{array}{l} \mathrm{E}_{\mathrm{NC}}\mathsf{msk}_1(\cdot), \\ \mathrm{E}_{\mathrm{NC}}\mathsf{msk}_2(\cdot), \\ \mathrm{KG}_{\mathrm{EN}\mathsf{msk}_1}(\cdot), \\ \mathrm{KG}_{\mathrm{EN}\mathsf{msk}_2}(\cdot) \end{array}\right)(1^\lambda)$
if $\mathsf{sk}_{f_1} \in \mathrm{L}_2 \vee \mathsf{sk}_{f_2} \in \mathrm{L}_1$:
return 0
if $\mathsf{Dec}(C, \mathsf{sk}_{f_1}) \neq \perp \wedge$
$\mathsf{Dec}(C, \mathsf{sk}_{f_2}) \neq \perp$:
return 1
return 0

$\underline{\mathrm{KG}_{\mathrm{EN}\mathsf{msk}_i}(f)}$:
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_i, f)$
$\mathrm{L}_i \leftarrow \mathrm{L}_i \cup \{(\mathsf{sk}_f, f)\}$
return sk_f

$\underline{\mathrm{E}_{\mathrm{NC}}\mathsf{msk}_i(M)}$:
$C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}_i, M)$
return C

$\underline{\text{FEROB}^{\mathcal{A}}_{\mathsf{PubFE}}(\lambda)}$:
$(\mathsf{mpk}_1, \mathsf{msk}_1, R_1, M_1, f_1, R_{f_1},$
$\mathsf{mpk}_2, \mathsf{msk}_2, R_2, M_2, f_2, R_{f_2}) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$C_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_1, M_1; R_1)$
$C_2 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_2, M_2; R_2)$
if $C_1 = C_2 \wedge \mathsf{mpk}_1 \neq \mathsf{mpk}_2$:
$\mathsf{sk}_{f_1} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_1, f_1; R_{f_1})$
$\mathsf{sk}_{f_2} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_2, f_2; R_{f_2})$
if $\mathsf{Dec}(C, \mathsf{sk}_{f_1}) \neq \perp \wedge$
$\mathsf{Dec}(C, \mathsf{sk}_{f_2}) \neq \perp$:
return 1
return 0

$\underline{\text{FEROB}^{\mathcal{A}}_{\mathsf{PrvFE}}(\lambda)}$:
$(\mathsf{msk}_1, R_1, M_1, f_1, R_{f_1},$
$\mathsf{msk}_2, R_2, M_2, f_2, R_{f_2}) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$C_1 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}_1, M_1; R_1)$
$C_2 \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}_2, M_2; R_2)$
if $C_1 = C_2 \wedge \mathsf{msk}_1 \neq \mathsf{msk}_2$:
$\mathsf{sk}_{f_1} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_1, f_1; R_{f_1})$
$\mathsf{sk}_{f_2} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}_2, f_2; R_{f_2})$
if $\mathsf{Dec}(C, \mathsf{sk}_{f_1}) \neq \perp \wedge$
$\mathsf{Dec}(C, \mathsf{sk}_{f_2}) \neq \perp$:
return 1
return 0

Figure 3.3: We introduce FEROB and SROB in the context of FE schemes defined both in the public and private key setting. For the SROB games, we give the oracles implementing Enc and KGen procedures, mentioning that each query to the latter oracle adds an entry of the form (f, sk_f) in the corresponding list L_i — where $i \in \{1, 2\}$ stands for the index of the used master keys.

Lemma 3.1 (Implications)**.** *Let* FE *denote a functional encryption scheme. If* FE *is* FEROB*-secure, then it is also* SROB*-secure.*

Lemma 3.1. We prove the implication holds in both the public and private key settings:
PUBLIC-KEY FE. We take the contrapositive. For a scheme FE, we assume the existence of an adversary $\mathcal{A}$ winning the SROB-game with non-negligible advantage ϵ_{SROB}. A reduction $\mathcal{R}$ that wins the FEROB game is built as follows: (1) $\mathcal{R}$ samples uniformly at random $(\mathsf{msk}_1, \mathsf{mpk}_1, \mathsf{msk}_2, \mathsf{mpk}_2)$; (2) the corresponding oracles for key-derivation are built; (3) $\mathcal{A}$ runs with access to the aforementioned oracles, returning $(C, \mathsf{sk}_{f_1}, \mathsf{sk}_{f_2})$. If $\mathcal{A}$ outputs a winning tuple, then $\mathcal{R}$ wins the FEROB game by releasing the messages and the randomness terms used to construct $(C, \mathsf{sk}_{f_1}, \mathsf{sk}_{f_2})$. Hence, $\mathsf{Adv}^{\text{srob}}_{\mathcal{A},\mathsf{FE}}(\lambda) \leq \mathsf{Adv}^{\text{ferob}}_{\mathcal{R},\mathsf{FE}}(\lambda)$.

PRIVATE-KEY FE. We take the contrapositive. For a scheme FE, we assume the existence of an adversary $\mathcal{A}$ winning the SROB-game with non-negligible advantage ϵ_{SROB}. A reduction $\mathcal{R}$ that wins the FEROB game is built as follows: (1) $\mathcal{R}$ samples uniformly at random $(\mathsf{msk}_1, \mathsf{msk}_2)$; (2) $\mathcal{R}$ constructs the encryption and key-derivation oracles under the two keys; (3) $\mathcal{R}$ runs $\mathcal{A}$ with access to these oracles, records the random coins used and obtains $(C, \mathsf{sk}_{f_1}, \mathsf{sk}_{f_2})$. Finally, $\mathcal{R}$ wins the FEROB game by issuing the FEROB tuple, using the random coins used to derive the functional keys and the ciphertext and therefore we have: $\mathsf{Adv}^{\text{srob}}_{\mathcal{A},\mathsf{FE}}(\lambda) \leq \mathsf{Adv}^{\text{ferob}}_{\mathcal{R},\mathsf{FE}}(\lambda)$. □

Proposition 3.1 (Separations)**.** *There exist functional encryption schemes in the public/private-key setting that are not* FEROB*-secure.*

Proposition 3.1. As sketched in Section 3.1, a DDH instantiation for the FE scheme of [ABDP15] is not FEROB-secure. The adversary is built upon the idea presented in the introduction and is shown in Figure 3.4. Given that any public-key functional encryption scheme can be trivially converted into one in the private-key setting simply by making mpk private, we obtain an FE scheme for the inner product functionality in the private-key setting that is not FEROB-secure.

FEROB adversary $\mathcal{A}^{\text{FEROB}}_{\mathsf{FE}}(\lambda)$:

1. $(g^{\vec{s}}, \vec{s}, r, \vec{x}, \vec{y}, \emptyset$
   $g^{\vec{s'}}, \vec{s'}, r, \vec{x'}, \vec{y}, \emptyset) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
   such that $r \cdot s_i + x_i = r \cdot s'_i + x'_i$ and $\vec{s} \neq \vec{s'}$
2. observe that $\mathsf{Enc}(g^{\vec{s}}, \vec{x}) = (g^r, g^{r \cdot s_1 + x_1} \ldots, g^{r \cdot s_n + x_n}) =$
   $(g^r, g^{r \cdot s'_1 + x'_1} \ldots, g^{r \cdot s'_n + x'_n}) = \mathsf{Enc}(g^{\vec{s'}}, \vec{x'})$
3. $\mathsf{sk}_y \leftarrow \vec{s}^\top \cdot \vec{y}$
4. $\mathsf{sk}'_y \leftarrow \vec{s'}^\top \cdot \vec{y}$
5. $\mathsf{Dec}(C, \mathsf{sk}_y) = \vec{y}^\top \cdot \vec{x} \neq \perp$
6. $\mathsf{Dec}(C, \mathsf{sk}'_y) = \vec{y}^\top \cdot \vec{x'} \neq \perp$

Figure 3.4: A FEROB adversary against the DDH instantiation of the bounded-norm inner product scheme in [ABDP15].

□

Intermediate Notions. Intermediate notions considering robustness under adversarially generated keys introduced in [FLPQ13] — such as full-robustness or mixed robustness — do not generalize well to functional encryption (or attribute-based encryption). The notion we consider, namely FEROB is, in fact, the generalization of KROB (key-less robustness), as introduced for PKE by Farshim et al. [FLPQ13].

3.2.2.2 Single-Authority Setting

The second stream of work would study the meaning of robustness in a single-authority context. In rough terms, the problem one would like to solve can be stated as: *if a ciphertext is correctly generated, and the adversary issues two functional keys, is there a chance that one of the keys fails in decrypting the ciphertext?* An astute reader may immediately notice that in such a setting, an adversary may always win such a game by issuing a pair of correct/random functional keys, as it owns the master secret key.

In a "dual" mode, if the functional keys are correctly generated under the same msk, *is there a ciphertext decryptable under one key and not under the other?* The intuition behind: if C is generated with respect to some mpk, we want the decryption to pass for any functional key correctly generated with respect to the $(\mathsf{mpk}, \mathsf{msk})$ pair. However, if C is obtained under some other $\mathsf{mpk}' \neq \mathsf{mpk}$ or is sampled according to some distribution, we expect decryption not to pass under any functional keys generated with respect to msk. Therefore, a set of potential meaningful definitions may capture this problematic case: decryption "works" under *one* correctly generated key out of two.

3.3 Generic Transforms

3.3.1 A Generic FEROB Transform in the Public-Key Setting.

As regards obtaining robust schemes generically, one can reuse the elegant idea rooted in the *binding* property of a commitment scheme. Concretely, one can start from an FE scheme, encrypt the plaintext, and post-process the resulting ciphertext through the use of a public-key encryption scheme. The transform consists in committing to the two public keys (corresponding to FE and PK) and encrypting the resulting decommitment together with the output of FE.Enc under pk. For decryption, in addition to the functional key, the secret key sk[2] is needed to recover the decommitment from the "middle" part of the ciphertext. A key difference to the ABN transform would be rooted in the innate nature of FE: one cannot encrypt the plaintext under pk, as this would break indistinguishability.

A simpler idea makes use of a collision-resistant hash function and simply appends the hash of $\mathsf{mpk}||C$ to the already existing ciphertext.

[2] sk is common to all users querying a sk_f.

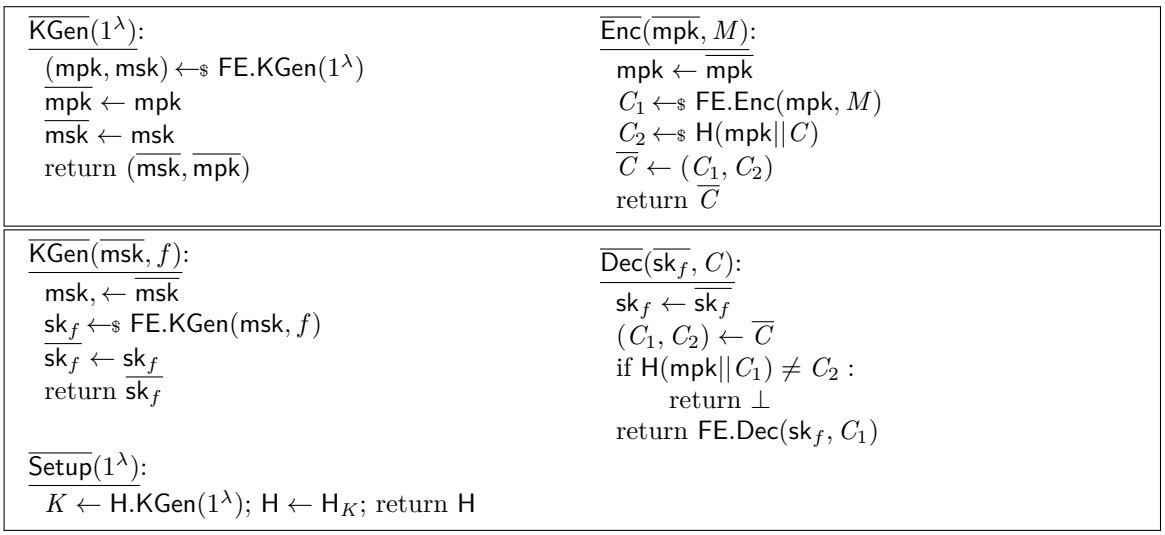

Figure 3.5: Generic transform that turns an FE scheme into a FEROB scheme $\overline{\mathsf{FE}}$.

Lemma 3.2. *Let* FE *be an* IND-FE-CPA*-secure functional encryption scheme in the public setting and let* H *denote a collision-resistant hash function. The functional encryption scheme* $\overline{\mathsf{FE}}$ *obtained through the transform depicted in Figure 3.5 is* FEROB*-secure, while preserving the* IND-FE-CPA*-security.*

Lemma 3.2. ROBUSTNESS. To show the transform achieves FEROB, we argue that if an adversary concludes with $(\overline{\mathsf{mpk}_1}, R_1, M_1, \overline{\mathsf{mpk}_2}, R_2, M_2, \ldots)$ such that $\overline{\mathsf{FE}}.\mathsf{Enc}(\overline{\mathsf{mpk}_1}, M_1; R_1) = \overline{\mathsf{FE}}.\mathsf{Enc}(\overline{\mathsf{mpk}_2}, M_2; R_2)$, then the adversary is essentially able to find two tuples such that $\mathsf{H}(\mathsf{mpk}_1||\mathsf{FE.Enc}(\mathsf{mpk}_1, M_1; R_1)) = \mathsf{H}(\mathsf{mpk}_2||\mathsf{FE.Enc}(\mathsf{mpk}_2, M_2; R_2))$ which cannot happen with non-negligible probability down to the collision-resistance of H.

INDISTINGUISHABILITY. The proof follows easily down to the indistinguishability of the underlying scheme FE: during the challenge phase, the reduction will be given the C^* corresponding to M_b (chosen by $\mathcal{A}$); after appending $\mathsf{H}(C^*||\mathsf{mpk})$, the adversary will be given $\overline{C^*}$. Observe that the reduction can answer all the functional key-derivation queries the adversary makes. Hence the advantage in winning the IND-FE-CPA game against $\overline{\mathsf{FE}}$ is bounded by the advantage of winning the IND-FE-CPA game against FE. □

3.3.2 Anonymity and Robustness

One can define the classical notion of anonymity to the context of functional encryption and its security experiment in Figure 3.6. We point out that usually, in an FE scheme, a central authority answers key-derivation queries from a potential set of users $\mathcal{U}$; therefore it is unnatural to assume that a user does not know from whom it received the functional key. What we want to ensure is that an adversary $\mathcal{A} \notin \mathcal{U}$ cannot tell *which* authority issued a ciphertext, without interacting with the key-derivation procedures; otherwise the game becomes trivial. In consequence, we define anonymity only in the context of public-key FE, as for a private scheme, the adversary uses encryption oracles to obtain a ciphertext. Thus, anonymity requires that a PPT bounded adversary can tell which mpk was used to encrypt a ciphertext only with negligible probability: $\mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A},\mathsf{FE}}(\lambda) \coloneqq 2 \cdot \Pr\left[\mathrm{ANON}^{\mathcal{A}}_{\mathsf{FE}}(\lambda) = 1\right] - 1 \in \mathrm{NEGL}(\lambda)$.

$\underline{\mathrm{ANON}^{\mathcal{A}}_{\mathsf{FE}}(\lambda)}$:
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{mpk}_0,\ \mathsf{msk}_0) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$(\mathsf{mpk}_1,\ \mathsf{msk}_1) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
$M \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, \mathsf{mpk}_0, \mathsf{mpk}_1)$
$C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_b, M)$
$b' \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, C)$
return $b = b'$

Figure 3.6: Anonymity for public-key functional encryption in the absence of functional keys.

Interestingly, FEROB does not imply anonymity for the public-key case. And based on FEROB $\Rightarrow$ SROB, it follows that SROB does not generically imply anonymity. Therefore, we have the following separation:

Proposition 3.2. *There exist* FEROB *transforms for public-key functional encryption that do not ensure anonymity (as defined in Figure 3.6).*

Proposition 3.2. We consider the scheme in Figure 3.5 and observe that the anonymity game can be easily won as follows: an adversary, given two master public keys and the ciphertext $\overline{C} \leftarrow (C_1, C_2)$, decides the issuer by checking whether $\mathsf{H}(C_1||\mathsf{mpk}_1) \stackrel{?}{=} C_2$ or $\mathsf{H}(C_1||\mathsf{mpk}_2) \stackrel{?}{=} C_2$, via the publicly available H. □

Remark 3.1. *A generic construction of an anonymous* FEROB *scheme, reaching both anonymity and robustness for* FE *is non-trivial: on the one hand, we expect the ciphertext to be "robust" w.r.t. a sole authority (*mpk*), but the "link" should not be detectable when included in the ciphertext (anonymity). Therefore, we attempt to embed such a link in the functional key. Our solution ensures* FEROB *through the means of a collision-resistant PRF (detailed in Section 3.4) with keys K generated on the fly. An independent functional key to compute the* PRF *value is issued via a second* FE *supporting general circuits, while the* PRF *key K is encrypted under the additional* mpk′.

Theorem 3.1. *Let* PRF *denote a collision-resistant* PRF *computable by circuits in a class $\mathfrak{C}$. Let* FE′ *be an* ANON*-secure functional encryption scheme supporting circuits in $\mathfrak{C}$. Given an* ANON, IND-FE-CPA*-secure scheme* FE*, the functional encryption scheme $\overline{FE}$ obtained via the transform in Figure 3.7 is* FEROB*-secure while preserving the original scheme's security guarantees.*

Theorem 3.1. ROBUSTNESS. FEROB follows from the collision resistance of the PRF: if an adversary $\mathcal{A}$ is able to find $(K, C_1), (K', C_1)$ such that $\mathsf{PRF}(K, C_1) = \mathsf{PRF}(K', C_1)$, then $\mathcal{A}$ wins the collision resistance game against the PRF.

INDISTINGUISHABILITY. Follows from the IND-FE-CPA-security of the underlying scheme FE. For any adversary $\mathcal{A}$ against the IND-FE-CPA-security of the scheme $\overline{\mathsf{FE}}$ in Figure 3.7, we build the reduction $\mathcal{R}$ that wins the IND-FE-CPA game against FE as follows:

First, the IND-FE-CPA experiment samples its own master keys and initializes the key-derivation oracle. The reduction $\mathcal{R}$ instantiates FE′ by sampling the master keys $(\mathsf{msk}', \mathsf{mpk}')$.

$\underline{\overline{\mathsf{KGen}}(1^\lambda)}$:
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow\!\!\$\ \mathsf{FE.KGen}(1^\lambda)$
$(\mathsf{mpk}', \mathsf{msk}') \leftarrow\!\!\$\ \mathsf{FE}'.\mathsf{KGen}(1^\lambda)$
$\overline{\mathsf{mpk}} \leftarrow (\mathsf{mpk}, \mathsf{mpk}')$
$\overline{\mathsf{msk}} \leftarrow (\mathsf{msk}, \mathsf{msk}')$
return $(\overline{\mathsf{msk}}, \overline{\mathsf{mpk}})$

$\underline{\overline{\mathsf{Enc}}(\overline{\mathsf{mpk}}, M)}$:
$(\mathsf{msk}, \mathsf{msk}') \leftarrow \overline{\mathsf{msk}}$
$(\mathsf{mpk}, \mathsf{mpk}') \leftarrow \overline{\mathsf{mpk}}$
$C_1 \leftarrow\!\!\$\ \mathsf{FE.Enc}(\mathsf{mpk}, M)$
$K \leftarrow\!\!\$\ \mathcal{K}$
$C_2 \leftarrow \mathsf{PRF}(K, \mathsf{mpk})$
$C_3 \leftarrow\!\!\$\ \mathsf{FE}'.\mathsf{Enc}(\mathsf{mpk}', K)$
$\overline{C} \leftarrow (C_1, C_2, C_3)$
return $\overline{C}$

$\underline{\overline{\mathsf{KGen}}(\overline{\mathsf{msk}}, f)}$:
$\mathsf{msk} \leftarrow \overline{\mathsf{msk}}$
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{FE.KGen}(\mathsf{msk}, f)$
$\mathsf{sk}_g \leftarrow\!\!\$\ \mathsf{FE}'.\mathsf{KGen}(\mathsf{msk}', \mathcal{C}_{\mathsf{PRF}(\cdot, \mathsf{mpk})})$
$\overline{\mathsf{sk}_f} \leftarrow (\mathsf{sk}_f, \mathsf{sk}_g)$
return $\overline{\mathsf{sk}_f}$

$\underline{\overline{\mathsf{Dec}}(\overline{\mathsf{sk}_f}, C)}$:
$(\mathsf{sk}_f, \mathsf{sk}_g) \leftarrow \overline{\mathsf{sk}_f}$
$(C_1, C_2, C_3) \leftarrow \overline{C}$
if $\mathsf{FE.Dec}(\mathsf{sk}_g, C_3) \neq C_2$:
    return $\perp$
return $\mathsf{FE.Dec}(\mathsf{sk}_f, C_1)$

Figure 3.7: A generic transform that converts an FE scheme into a FEROB scheme $\overline{\mathsf{FE}}$, *without* ensuring anonymity. Here $\mathcal{C}_{\mathsf{PRF}}$ denotes the circuit that computes the PRF value, where mpk is hard-coded in the circuit.

Regarding the challenge ciphertext, whenever the adversary $\mathcal{A}$ sends the challenge tuple (M_0, M_1), the reduction $\mathcal{R}$ proceeds as follows: (1) obtains challenge ciphertext C_1 from the IND-FE-CPA experiment; (2) samples (on the fly) its own key K; (3) computes C_2, C_3, which are forwarded to $\mathcal{A}$. Note that all these steps are perfectly computable, as $\mathcal{R}$ knows mpk'.

Regarding key-derivation queries, whenever $\mathcal{A}$ requests a functional key for some f, $\mathcal{R}$ forwards the request to the key-generation oracle. Independently, the reduction obtains a functional key for $\mathcal{C}_{\mathsf{PRF}(\cdot,\mathsf{mpk})}$, a circuit that is designed to compute C_2 (the PRF value) over the encrypted K.

It is clear the reduction $\mathcal{R}$ can simulate the IND-FE-CPA game for $\overline{FE}$ in the view of its adversary $\mathcal{R}$. Thus, whenever $\mathcal{A}$ returns b, $\mathcal{R}$ returns the same bit and wins under the same advantage.

ANONYMITY. Follows from the anonymity of the underlying FE scheme. We use a hybrid argument. We start from a setting corresponding to $b = 0$ in the $\mathrm{ANON}^{\mathcal{A}}_{\overline{\mathsf{FE}}}$ game (Game_0).

- Game_1**:** in Game_1, we change C_3 from $\mathsf{FE}'.\mathsf{Enc}(\mathsf{mpk}_0, K)$ to $\mathsf{FE}'.\mathsf{Enc}(\mathsf{mpk}_1, K)$, based on the ANON property of FE', the hop between the two games being bounded by $\mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A},\mathsf{FE}'}(\lambda)$.

- Game_2**:** we change C_1 from $\mathsf{FE.Enc}(\mathsf{mpk}_0, M)$ to $\mathsf{FE.Enc}(\mathsf{mpk}_1, M)$, based on the anonymity of the underlying FE scheme, the distance to the previous game being bounded by $\mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A},\mathsf{FE}}(\lambda)$. Implicitly, in Game_2, the reduction updates the value of the PRF from $\mathsf{PRF}(K, \mathsf{FE.Enc}(\mathsf{mpk}_0, C_1))$ to $\mathsf{PRF}(K, \mathsf{FE.Enc}(\mathsf{mpk}_1, C_1))$.

Finally, observe that Game_2 maps to the setting where $b = 1$ in the anonymity game for the $\overline{\mathsf{FE}}$ scheme. Therefore, $\mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A},\overline{\mathsf{FE}}}(\lambda) \leq \mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A}_1,\mathsf{FE}'}(\lambda) + \mathsf{Adv}^{\mathrm{anon}}_{\mathcal{A}_2,\mathsf{FE}}(\lambda)$. □

$\overline{\mathsf{KGen}}(1^\lambda)$:
$R \leftarrow\!\!\$\ \{0,1\}^\lambda$
$R_1 || R_2 \leftarrow \mathsf{PRG.Eval}(R)$
$\mathsf{msk} \leftarrow \mathsf{FE.Enc}(1^\lambda; R_1)$
$\mathsf{sk} \leftarrow R_2$
$\overline{\mathsf{msk}} \leftarrow (\mathsf{msk}, \mathsf{sk})$
return $\overline{\mathsf{msk}}$

$\overline{\mathsf{Enc}}(\overline{\mathsf{msk}}, M)$:
$(\mathsf{msk}, \mathsf{sk}) \leftarrow \overline{\mathsf{msk}}$
$C_1 \leftarrow\!\!\$\ \mathsf{FE.Enc}(\mathsf{msk}, M)$
$C_2 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(\mathsf{sk}, C_1)$
$\overline{C} \leftarrow (C_1, C_2)$
return $\overline{C}$

$\overline{\mathsf{KGen}}(\overline{\mathsf{msk}}, f)$:
$(\mathsf{msk}, \mathsf{sk}) \leftarrow \overline{\mathsf{msk}}$
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{FE.KGen}(\mathsf{msk}, f)$
$\overline{\mathsf{sk}_f} \leftarrow (\mathsf{sk}_f, \mathsf{sk})$
return $\overline{\mathsf{sk}_f}$

$\overline{\mathsf{Dec}}(\overline{\mathsf{sk}_f}, C)$:
$(\mathsf{sk}_f, \mathsf{sk}) \leftarrow \overline{\mathsf{sk}_f}$
$(C_1, C_2) \leftarrow \overline{C}$
if $\mathsf{PRF.Eval}(\mathsf{sk}, C_1) \neq C_2$:
    return $\perp$
return $\mathsf{FE.Dec}(\mathsf{sk}_f, C_1)$

Figure 3.8: A generic transform that turns a FE scheme in the private-key setting into a FEROB-secure scheme $\overline{\mathsf{FE}}$.

3.3.3 FEROB Transform in the Private-Key FE Setting.

In this part, we provide a similar generic transform for turning any FE scheme into one that is FEROB-secure, in the private-key framework.

Lemma 3.3. *Let* FE *be an* IND-FE-CPA *functional encryption scheme in the private-key setting. Let* PRG *denote a right-injective length doubling pseudorandom generator from* $\{0,1\}^\lambda$ *to* $\{0,1\}^{2\cdot\lambda}$ *and* PRF *a collision-resistant* PRF*. The functional encryption scheme* $\overline{\mathsf{FE}}$ *obtained through the transform depicted in Figure 3.8 is* FEROB*-secure, while preserving* IND-FE-CPA*-security.*

Lemma 3.3. ROBUSTNESS. Assuming the FEROB adversary $\mathcal{A}$ outputs $(\overline{\mathsf{msk}_1}, R_1, M_1, f_1, R_{f_1}, \overline{\mathsf{msk}_2}, R_2, M_2, f_2, R_{f_2})$ such that $\overline{\mathsf{FE}}.\mathsf{Enc}(\overline{\mathsf{msk}_1}, M_1; R_1) = \overline{\mathsf{FE}}.\mathsf{Enc}(\overline{\mathsf{msk}_2}, M_2; R_2)$, we argue that:

- $C_2 = \mathsf{PRF.Eval}(\mathsf{sk}_1, C_1) = \mathsf{PRF.Eval}(\mathsf{sk}_2, C_1)$. Down to the collision-resistance (over both keys and inputs) property of the PRF, it results that $\mathsf{sk}_1 = \mathsf{sk}_2$.
- the $\overline{\mathsf{KGen}}$ function makes use of a right injective pseudorandom generator. Since the right half is exactly $\mathsf{sk}_1 (= \mathsf{sk}_2)$, through the injectivity property, it must be the case that the seed R used to feed the PRG is the same.
- since the randomness R is the same for both cases, it results that the random coins used by FE.KGen are the same, implying that $\mathsf{msk}_1 = \mathsf{msk}_2$.
- finally, we obtain that $\overline{\mathsf{msk}_1} = \overline{\mathsf{msk}_2}$, which is not allowed in the robustness game.

Therefore, the advantage of breaking the FEROB game is bounded by the union bound applied on the collision-resistance of the PRF and right-injectivity of the PRG: $\mathsf{Adv}^{\mathrm{ferob}}_{\mathcal{A},\overline{\mathsf{FE}}}(\lambda) \leq \mathsf{Adv}^{\mathrm{inj}}_{\mathcal{R},\mathsf{PRG}}(\lambda) + \mathsf{Adv}^{\mathrm{cr}}_{\mathcal{R}',\mathsf{PRF}}(\lambda)$.

IND-FE-CPA-SECURITY. The reduction proceeds via one game hop:

- Game_0: is the game, where the adversary runs against the scheme depicted in Figure 3.8 — the output of the PRG is the expected one.
- Game_1: based on the pseudorandomness property of the PRG, we change the output to a truly random string, ensuring independence between msk and sk. The distance to Game_0 is bounded by the pseudorandomness advantage against PRG. We now show that the advantage of an adversary winning the IND-FE-CPA experiment against $\overline{\mathsf{FE}}$ in this setting is negligible.

Assume the existence of a PPT adversary $\mathcal{A}$ against the IND-FE-CPA of $\overline{\mathsf{FE}}$. We build an adversary $\mathcal{R}$ against the IND-FE-CPA of the underlying FE scheme. The IND-FE-CPA experiment samples a bit b', the key msk and constructs a key-derivation oracle KGen under msk, such that it can be accessed $\mathcal{R}$. The reduction then proceeds as follows:

1. $\mathcal{R}$ chooses uniformly at random sk to key the PRF utility.
2. $\mathcal{R}$ builds the $\overline{\mathsf{FE}}.\mathsf{Enc}$ oracle and the $\overline{\mathsf{FE}}.\mathsf{KGen}$ oracle by querying the given $\mathsf{FE}.\mathsf{Enc}, \mathsf{FE}.\mathsf{KGen}$. The PRF is evaluated under sk.
3. $\mathcal{R}$ runs $\mathcal{A}$, obtains a tuple (M_0, M_1) and gets back the encryption of $M_{b'}$ (say C^*) by querying $\mathsf{FE}.\mathsf{Enc}(\mathsf{msk}, M_{b'})$. $\mathcal{R}$ computes the corresponding $\overline{C^*}$, which is passed to $\mathcal{A}$.
4. finally, $\mathcal{A}$ returns a bit b, which constitutes the output of $\mathcal{R}$.

Analysis of the reduction. The correctness of the reduction follows trivially. Thus we conclude that in Game_1, the probability of winning is:

$$\Pr[\text{Game}_1^{\mathcal{A}}] \leq \mathsf{Adv}^{\text{ind-fe-cpa}}_{\mathcal{R},\mathsf{FE}}(\lambda) \ .$$

For the analysis, we also include the fact that the transition between Game_0 and Game_1 is bounded by the pseudorandomness of PRG:

$$\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}] \leq \mathsf{Adv}^{\text{prg}}_{\mathcal{R}',\mathsf{PRG}}(\lambda) \ .$$

Finally, it follows that:

$$\mathsf{Adv}^{\text{ind-fe-cpa}}_{\mathcal{A},\overline{\mathsf{FE}}}(\lambda) \leq \mathsf{Adv}^{\text{ind-fe-cpa}}_{\mathcal{R},\mathsf{FE}}(\lambda) + \mathsf{Adv}^{\text{prg}}_{\mathcal{R}',\mathsf{PRG}}(\lambda) \ .$$

□

3.4 Robust and Collision-Resistant PRFs

In this section, we show how to construct the main ingredient needed in the aforementioned generic transform: collision-resistant PRFs.

3.4.1 Definitions

Collision resistance for PRFs. As stated in Chapter 2, pseudorandom functions are basic primitives having the output distribution indistinguishable from the uniform distribution. This happens as long as its key remains secret. Naturally, we expect a PRF to reach high

$\underline{\text{FROB}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda)}$:
$(K_1, M_1, K_2, M_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
if $K_1 = K_2$ return 0
$T_1 \leftarrow \mathsf{Eval}(K_1, M_1)$
$T_2 \leftarrow \mathsf{Eval}(K_2, M_2)$
return $(T_1 = T_2)$

Figure 3.9: Game defining full robustness for a pseudorandom function PRF.

guarantees of collision resistance property assuming the secrecy of K. For the sake of clarity, we define collision resistance of a pseudorandom function PRFs as:

$$\mathsf{PRF}(K_1, M_1) = \mathsf{PRF}(K_2, M_2) \implies (K_1, M_1) = (K_2, M_2) \ .$$

However, things change dramatically as soon as an adversary starts tampering with the key. Consider for instance the simple PRF by Naor and Reingold, where the output is modelled as:

$$\mathsf{PRF}(K, M) = g^{a_0 \cdot \prod_{i=1}^{n} a_i^{M_i}}$$

Whenever the adversary is given the ability to tamper with the key generation, it may easily craft the key $K = \mathbf{a}$ such that for two specific messages $\mathsf{PRF}(K, M_1) = \mathsf{PRF}(K, M_2)$.

In the forthcoming parts, we delve into the constructions of PRFs achieving collision resistance. Such constructions are accomplished by combining: (1) length-doubling right-injective PRGs and (2) key-injective PRFs. The latter primitive can be obtained via the GGM construction (see also [CHN+16, Appendix C]).

Robustness for PRFs. We also define a slight variation of the robustness game in the context of PRFs in Figure 3.9. In such sense, this game would correspond to the collision-resistance property, but under the additional constraint that the adversary is required to produce a pair of differing keys ($K_1 \neq K_2$).

Definition 3.2 (Robustness for PRFs)**.** *We say a pseudorandom function* PRF *is robust if the advantage of any* PPT *adversary in winning the game in Figure* 3.9

$$\mathsf{Adv}^{\text{frob}}_{\mathcal{A},\mathsf{PRF}}(\lambda) := \Pr\left[\text{FROB}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda) = 1\right] \ .$$

is negligible.

As we shall see, from a foundational perspective, robust PRFs underlie feasibility of robustness for many symmetric primitives.

3.4.2 Construction of Robust and Collision-Resistant PRFs

We now turn to the problem of constructing robust and collision-resistant PRFs. For practical purposes, it is a reasonable assumption that a keyed hash function acts as a PRF when used with a random and unknown key, and is also an unkeyed collision-resistant function.[3] Hence, a practical hash function can be used to instantiate the transformations in the previous section.

[3] And indeed the random oracle meets this simultaneous security requirement.

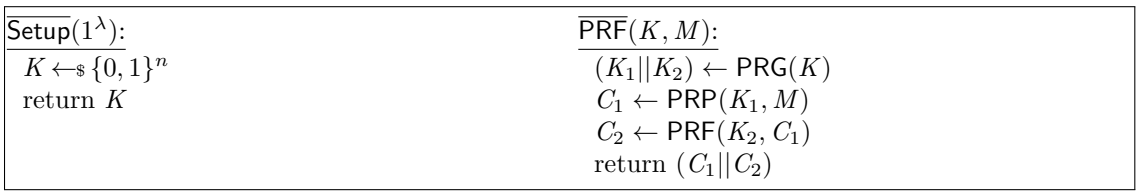

Figure 3.10: Collision-resistant PRF from a key-injective PRF. Keys are derived via a right-injective length-doubling PRG.

We ask if collision-resistant PRFs can be based on simpler assumptions in the standard model. One method to immediately obtain collision-resistant PRFs would be to use *combiners*. Roughly speaking, a hash function combiner is a transform that takes two (or more) hash functions as input and outputs a hash function that is secure if either a hash function is secure. For example, concatenation is a combiner for collision resistance. Fischlin et al. [FLP14] give a multi-property combiner for hash functions that is proved to *simultaneously* preserve multiple security properties of its input hash functions, including collision-resistance and pseudorandomness; this raises an alternative route to obtain collision-resistant/robust PRFs based on multi-property hash combiners. The construction of Fischlin et al. [FLP14], however, considers keyed collision resistance which is not sufficient for our purposes. Furthermore, a modification to unkeyed hash functions results in key dependency issues (somewhat similarly to the ABN transform) which then prevents a security proof.

Our first result of this section is a simple transform that converts any CROB-secure PRF into a fully collision-resistant PRF. In this transform, which is shown in Figure 3.10, we use a length-doubling PRG that is collision resistant on the right half of its output. We expand a key K to $(K_1||K_2)$ via a PRG, use K_2 in a *key-injective* PRF and K_1 in a pseudorandom *permutation* to guarantee collision resistance over both keys and inputs. *Key-injective* PRF [CMR98; Fis99] is a weakening of FROB where it is required that $M_1 = M_2$, i.e., it should be infeasible to find $K_1 \neq K_2$ such that $\mathsf{PRF}(K_1, M) = \mathsf{PRF}(K_2, M)$. We will also use a pseudorandom permutation PRP to ensure injectivity over messages.

Proposition 3.3. *The PRF construction in Figure 3.10 is collision-resistant (and in particular* CROB*) if the underlying* PRF *is key-injective and the* PRG *is right collision-resistant. Furthermore, the construction is* PRF *secure if the* PRG*,* PRF*, and* PRP *are secure.*

Proposition 3.3. We first prove collision resistance. Suppose an adversary outputs $(K, M) \neq (K', M')$ such that $\overline{\mathsf{PRF}}(K, M) = \overline{\mathsf{PRF}}(K', M')$. Let $(K_1, K_2) \leftarrow \mathsf{PRG}(K)$ and similarly let $(K'_1, K'_2) \leftarrow \mathsf{PRG}(K')$. Then by construction:

$$\mathsf{PRF}(K_2, C) = \mathsf{PRF}(K'_2, C), \quad \text{where} \quad C = \mathsf{PRP}(K_1, M) = \mathsf{PRP}(K'_1, M')$$

This means that the adversary breaks the assumed key-injectivity property of the PRF unless $K_2 = K'_2$ (note that the PRF is run on the same input). But, $K_2 = K'_2$ implies that we also have $K = K'$ as otherwise, the adversary would break the right collision-resistance property of the PRG. This, however, means that $K_1 = K'_1$. Now since PRP is a permutation over this key, collisions can only occur if $M = M'$. This, however, contradicts the assumption that $(K, M) \neq (K', M')$. The proof of PRF security is standard and proceeds as follows.

Game_0 : This is the PRF experiment with $b = 0$, the outputs being computed using the $\overline{\mathsf{PRF}}$.

Game_1 : In this game, instead of outputs of PRG we use random and independent K_1 and K_2. The distance to the previous game can be bounded via the security of PRG. This step decouples the two keys.

Game_2 : In this game, we replace the outputs of the PRF with random strings. The distance to the previous game can be bounded via the PRF security of the PRF.

Game_3 : In this game, we replace the outputs of the PRP with random strings. The distance to the previous game can be bounded via the security of the PRP. This game corresponds to PRF experiment with $b = 1$.

Therefore, for any $\mathcal{A}$ there are $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{B}_3$ such that

$$\mathsf{Adv}^{\text{prf}}_{\overline{\mathsf{PRF}},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{prg}}_{\mathsf{PRG},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\text{prf}}_{\mathsf{PRP},\mathcal{B}_3}(\lambda) \ .$$

□

We now prove that the key-injective PRF used above can be based on length-doubling PRGs that achieve collision-resistance both on the left and the right halves of their outputs. That is, for any efficient $\mathcal{A}$ the probability

$$\Pr\left[(K^1, K^2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda); (K_0^i, K_1^i) \leftarrow \mathsf{PRG}(K^i);\ \text{return}\ (K_0^1 = K_0^2 \vee K_1^1 = K_1^2) \wedge K^1 \neq K^2\right]$$

is negligible. We call such a PRG *left-right collision-resistant* (LRCR). The next lemma build on results from [CMR98; Fis99] shows that the GGM construction [GGM86], when instantiated with an LRCR-secure PRG is key-injective. Recall that the GGM construction defines a PRF as

$$\mathsf{PRF}(K, [M_0, \ldots, M_n]) := \mathsf{PRG}_{M_n}(\mathsf{PRG}_{M_{n-1}}(\ldots \mathsf{PRG}_{M_1}(K)\ldots)) \ ,$$

where M_i denotes the i-th bit of M, $\mathsf{PRG}_0(K)$ the left half of the output of $\mathsf{PRG}(K)$ and $\mathsf{PRG}_1(K)$ its right half. The difference with [CMR98; Fis99] is that we do not rely on a CRS (a.k.a. tribe-key) but on the stronger LRCR security of the PRG.

Proposition 3.4. *The GGM construction, when instantiated with a left/right collision-resistant* PRG*, results in a key-injective pseudorandom function.*

Proposition 3.4. The pseudorandomness proof is identical to that of the GGM. We prove key-injectivity. Let

$$y_j^i = \mathsf{PRG}_{M_i}(\mathsf{PRG}_{M_{i-1}}(\ldots \mathsf{PRG}_{M_1}(K_j)\ldots))$$

be the i-th intermediate value for key j. Suppose an adversary finds $(K_1, K_2, M = [M_1, \ldots, M_n])$ with $K_1 \neq K_2$ such that $y_1^n = y_2^n$. Now either $y_1^{n-1} \neq y_2^{n-1}$ or $y_1^{n-1} = y_2^{n-1}$. In the first case a collision is found, and we are done. In the second case we look at y_1^{n-2} and y_2^{n-2} and so on. If we reach y_1^1 and y_2^1 and a collision is yet to be found then, since $K_1 \neq K_2$, this is the collision for the PRG. □

Finally, we show that left/right collision-resistant PRGs can be built in the standard model (without the use of ROs). Consider the function $G : \mathbb{Z}_p^3 \longrightarrow \mathbb{G}^6$ for a group $\mathbb{G}$ of order p generated by g [BCP02]:

$$G(x_1, x_2, x_3) := (g^{x_1}, g^{x_1 x_2}, g^{x_2 x_3}, g^{x_2}, g^{x_1 x_3}, g^{x_3}) \ .$$

$\overline{\mathsf{PRG}}(s)$:
$(a_0, b_0, a_1, b_1, x) \leftarrow \mathsf{PRG}_0(s)$
if $(a_0 = 0 \vee a_1 = 0)$ then return $\perp$
$(x_0, x_1) \leftarrow G(x)$
$s_0 \leftarrow \mathsf{H}((a_0, b_0)||\pi((a_0, b_0), x_0))$; $s_1 \leftarrow \mathsf{H}((a_1, b_1)||\pi((a_1, b_1), x_1))$
return $s_0||s_1$

Figure 3.11: A length-doubling left/right collision resistant $\overline{\mathsf{PRG}} : \{0,1\}^n \to \{0,1\}^{2n}$, based on a regular collision-resistant hash $\mathsf{H} : \{0,1\}^{3\cdot 3(n+l)} \to \{0,1\}^n$, a pairwise-independent permutation $\pi : \{0,1\}^{3(n+l)} \to \{0,1\}^{3(n+l)}$, a LRCR-secure PRG $G : \{0,1\}^{3n} \to \{0,1\}^{2\cdot 3(n+l)}$ and a $\mathsf{PRG}_0 : \{0,1\}^n \to \{0,1\}^{4\cdot 3(n+l)+3n}$ right injective over the last $3n$ bits.

We start by observing that this function is indeed injective on its left and right halves of output. Suppose there exists $(x_1, x_2, x_3) \neq (y_1, y_2, y_3)$ such that $(g^{x_1}, g^{x_1x_2}, g^{x_2x_3}) = (g^{y_1}, g^{y_1y_2}, g^{y_2y_3})$. Then by comparing the first elements, we must have $x_1 = y_1$, which in conjunction with the equality of second components implies $x_2 = y_2$; this together with the equality of third components implies $x_3 = y_3$. Injectivity for the right half of the outputs is shown similarly.

The outputs of G, when running on random inputs are indistinguishable from a random element of $\mathbb{G}^6$ under the DDH assumption. For clarity of exposition, we start with $(g^{x_1}, g^{x_1x_2}, g^{x_2x_3}, g^{x_2}, g^{x_1x_3}, g^{x_3})$ and replace $g^{x_1x_2}$ with g^{z_1} using DDH applied to $(g^{x_1}, g^{x_2}, g^{x_1x_2})$ and generating an x_3 to simulate the remaining elements. Next, we replace $g^{x_2x_3}$ with g^{z_2} via DDH applied to $(g^{x_2}, g^{x_3}, g^{x_2x_3})$ and generate an x_1 to simulate the remaining elements. We finally replace $g^{x_1x_3}$ with g^{z_2} using DDH.

The outputs of G, however, are not GGM-friendly as they lie in $\mathbb{G}$ which may be encoded as strings that are longer than $2 \cdot |(x_1, x_2, x_3)|$. Furthermore, these outputs are not uniformly distributed. Instead, the outputs are indistinguishable from some distribution $\mathcal{D} \times \mathcal{D}$ on $\{0,1\}^{3(n+l)} \times \{0,1\}^{3(n+l)}$, where l is the length of the bits needed to represent the group elements.

Following [Dod05; DS05], we address these issues by applying in parallel a *collision-resistant extractor* to the outputs of G in two steps: (1) we apply a pairwise-independent permutation to bring the output distribution close to uniform; (2) we then use a collision-resistant, regular hash function to compress the result down to n bits without losing uniformity of the outputs. A pairwise-independent permutation π can be instantiated as

$$\pi((a, b), X) := a \cdot X + b \quad \text{where} \quad a, b \leftarrow_{\$} \{0,1\}^{3(n+l)}, a \neq 0$$

(where the $\cdot$ and $+$ operations are defined over an extension field). A function $\mathsf{H} : D \longrightarrow R$ is regular if its outputs are uniformly distributed over R for uniform inputs in D, equivalently for all $y \in R$ it holds $|\mathsf{H}^{-1}(y)| = |D|/|R|$. Regular, collision-resistant hash functions can be obtained from claw-free permutations [CMR98].

We define the required LRCR-secure and GGM-friendly $\overline{\mathsf{PRG}}$ in Figure 3.11, where PRG_0 is a right-injective PRG and $G(x) = (x_0, x_1)$ is an LRCR-secure PRG (for example the one above obtained from DDH).

Theorem 3.2. *The $\overline{\mathsf{PRG}}$ in Figure 3.11 is* LRCR*-secure and a secure PRG if PRG_0 is secure, G is secure with respect to the output distribution of $\mathcal{D}$ with min-entropy at least $3n$, H is a regular and collision-resistant hash function, and π is a pairwise-independent permutation.*

Theorem 3.2. We first show $\overline{\mathsf{PRG}}$ is LRCR secure. Let $\overline{\mathsf{PRG}}(s) = s_0||s_1$. Suppose that an adversary outputs $s \neq s'$ such that $s_d = s'_d$ for some $d = 0, 1$. Let $d = 0$. So either the adversary can be used to break the collision resistance of H or $((a_0, b_0), \pi((a_0, b_0), x_0)) = ((a'_0, b'_0), \pi((a'_0, b'_0), x'_0))$. Therefore $(a_0, b_0) = (a'_0, b'_0)$ and $\pi((a_0, b_0), x_0) = \pi((a'_0, b'_0), x'_0)$. Since $\pi((a_0, b_0), \cdot)$ is a permutation we must have that $x_0 = x'_0$. This contradicts the LRCR security of G unless $x = x'$. This in turns means that a collision on the right side (corresponding to x) of the output of PRG_0 is found unless $s = s'$. The case $d = 1$ is dealt with similarly. This concludes the proof of LRCR security.

We now turn to the pseudorandomness of the $\overline{\mathsf{PRG}}$. If H is regular, its outputs are uniform when fed with uniform inputs. Hence, we show that the outputs of π are uniform. We prove this by first replacing the key (a_0, b_0) (and respectively, (a_1, b_1)) of π with truly random keys using the security of PRG_0. We then replace x_0 (and respectively x_1) with random strings sampled according to the distribution $\mathcal{D}$ on $\{0,1\}^{3(n+l)}$; this follows from the security of G. Note that the distribution $\mathcal{D}$ has min-entropy at least $3n$ by the injectivity of group exponentiation.

Dodis and Smith [DS05, Prop. 11] show a left-over hash lemma for composition with functions: for H a regular collision-resistant hash function with output length $\ell \leq t - 2\log(\frac{1}{\epsilon})$, where t is the min-entropy of the input source $\mathcal{D}$ to a pairwise-independent permutation π, the statistical distance between $\mathsf{H}(\pi(\mathcal{D}))$ and $\mathsf{H}(\mathcal{U})$ is at most ϵ. Applying this result to our setting with $\epsilon := 2^{-n}$, we get that setting $\ell \leq 3n - 2\log(\frac{1}{\epsilon}) = n$ would result in uniform outputs; this matches the output length of H and concludes the proof of security of $\overline{\mathsf{PRG}}$. □

Remark. We note that LRCR security is also necessary for building key-injective PRFs as any key-injective PRF would immediately give rise to an LRCR-secure PRG by setting the seed to the PRF key and the outputs of the PRG to those of the PRF evaluated at two points. We leave the possibility of basing LRCR-secure PRGs on generic assumptions, such as one-way functions/permutations or collision-resistance, to future work. We, however, observe that collision-resistance does not seem to be a necessary condition as the left or right halves of the PRG do not need to be compressing.

Chapter 4

Robust Authentication

In this chapter, we define robustness for authentication primitives, focusing on message authentication codes, authenticated encryption and digital signatures schemes. We show how to obtain generic transforms for the aforementioned primitives in both the random oracle model and the standard model. Throughout this chapter, we make extensive use of the techniques we introduce in Chapter 3, relying chiefly on the collision resistance of pseudorandom functions.

4.1 Robustness for Authentication Primitives

The vast and diverse landscape of cryptographic notions of security is a useful resource for the academic community, as it allows to describe, precisely, what kind of properties a certain cryptographic scheme guarantees – and implicitly which one it does not. However, this complexity often hinders the ability of practitioners and users of cryptography to implement genuinely secure cryptographic systems. In the eyes of the users, cryptography is often seen as an *all-or-nothing* process: once cryptography is "turned on", data gets encrypted, and therefore the system is secure, hopefully with as little fine print as possible. The shortcomings of this all-or-nothing property have been shown by a long series of attacks on real-world cryptographic protocols.

The academic community is reacting to this real-world need with more straightforward and more comprehensive notions of security. The clearest example of this is the introduction of the notion of *Authenticated Encryption* (AE) [Rog02; RS06]. While early cryptography considered confidentiality to be the only goal of encryption, over the years, it has become apparent that virtually every application requiring confidentiality would also benefit from some form of authenticity guarantees. Therefore, instead of letting the users pick an encryption and a MAC scheme (and combine them in appropriate ways), cryptographers are currently designing schemes that guarantee all properties at once (cf. the CAESER competition). Other examples in this direction are the study of *misuse-resistant AE schemes* [RS06], which guarantee the best possible security even in the presence of repeating nonces, security under related-key attacks (RKAs) [Bih94; BK03], and security in the presence of key-dependent messages [BRS02].

In this quest towards coming up with AE schemes that are as *ideally secure* as possible, we extend the notion of *key-robustness*[1]. In a nutshell, key-robustness looks at a setting where multiple keys (possibly known and/or chosen by the adversary) are present in the system. When using strong encryption, like authenticated encryption, it might be tempting to assume that any given ciphertext would only be valid for a single secret key. As we shall see, this may or may not be the case depending on the context. We start with some motivating examples before discussing the details.

Example 1 – Storage Authenticity: In this application, a user wants to encrypt some data which is stored on an untrusted storage provider. To ensure the authenticity of the data, the user encrypts it using an AE scheme. Then the user stores the key in clear on a different storage provider. What happens now if the second storage provider is corrupt? It might be tempting to think that, since the data is encrypted with AE, any tampering on the key will be detected when the user decrypts the data with the key. Unfortunately, this is not the case, and as we discuss in Section 4.3, AE security alone does not guarantee the authenticity of the original data against an adversary that can tamper with the stored key. Note that tampering with the key can be done with the knowledge of the original key.

Example 2 – Oblivious Transfer (OT): Consider the following protocol, for constructing a $\binom{3}{2}$-OT protocol using only $\binom{3}{1}$-OTs: the sender picks 3 random keys k_1, k_2, k_3 and inputs the message $x_1 = (k_2, k_3), x_2 = (k_1, k_3)$ and $x_3 = (k_1, k_2)$ to the OT. At the same time, the

[1] We refrain from *formally* referring to this notion as robustness in order to avoid confusion with robust AE schemes [HKR15]. In our discussions, however, we use robustness to ease readability.

sender sends encryptions of his messages under these keys, i.e., sends $c_i = E(k_i, m_i)$ for $i = 1..3$. Now the receiver inputs the index of the message *he does not want to learn* to the $\binom{3}{1}$-OT and learns all keys except k_i. Intuitively the fact that the messages are sent only once (encrypted) should guarantee that the sender's choice of messages is uniquely defined. However, consider the following attack: the corrupt sender inputs $x_1^* = (k_2, k^*)$ (instead of x_1) such that $D(k^*, c_3) = m_3^*$ with $m_3^* \neq m_3$ and $m_3^* \neq \perp$; this means that the receiver will see two different versions of m_3 depending on whether the receiver asked for the pair $(2, 3)$ or $(1, 3)$. (This attack is an example of *input-dependence* and is a clear breach of security since it cannot be simulated in the ideal world.) The attack described here is a simplified version of an actual attack described by [Lam16] on the private set-intersection protocol of [DCW13]. A strong form of key-robustness for symmetric encryption is also used to prove the security of the OT protocol presented in [CO15].

Example 3 – Signature Schemes: Digital signature schemes are used to authenticate electronic documents publicly. The textbook notion, capturing the *existential unforgeability* of a DS ensures that an adversary, interacting with *one* signing oracle, cannot forge a signature (for a message he did not previously query). On the other hand, a real-world scenario is placed in a multi-user context, where it is often assumed (but not necessarily proven) that a signature can verify *only* under the issuer's key.

Consider a practical situation with a clerk *acquiring* a digital signature for daily use, a third-party generating the pairs of keys. Even if the scheme remains unforgeable according to the classical definition, we do not have formal guarantees that two pairs of keys — $(\mathsf{sk}, \mathsf{pk})$ and $(\mathsf{sk}', \mathsf{pk}')$ — generated by the third party (potentially *maliciously*), cannot be used to produce a signature σ for some *chosen message* M, verifiable under both pk and pk' — something completely undesirable in practice. To be fully explicit with our example, suppose one pair of keys $(\mathsf{pk}, \mathsf{sk})$ is given to the clerk and a second one $(\mathsf{pk}', \mathsf{sk}')$, is issued by the third-party and is covertly used by a local/global security agency. When needed (and if needed), an operator can issue a signature (using sk') for the message: "I attest XYZ is true." which can later be verified under pk, thus having baleful consequences for the clerk.

To give a taste of a signature scheme where such an attack is feasible, consider the one obtained from a toy version of the Boneh–Boyen scheme [BB04]. The construction is *pairing*-based and can be summarized as follows: (1) key-generation samples two group generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, both of order p, and publishes as a public key $(g_1, g_2, g_2^x, e(g_1, g_2))$ — for a uniformly sampled $x \in \mathbb{Z}_p$ — keeping x as a secret key. To sign the message M, one computes $\sigma \leftarrow g_1^{1/(x+M)}$. A robustness attack against this simple signature scheme exploits the randomness in choosing the secret keys, observing that for a different pair $(\mathsf{pk}', \mathsf{sk}')$, one can select $g_1' \leftarrow g_1^t$ and then can set $x' \leftarrow t(x+M) - M$ (over $\mathbb{Z}_p$) such that $\sigma \leftarrow {g'}^{1/(x'+M)}$.

4.1.1 Chapter Organization

We give simple and strong definitions of key-robustness for a number of symmetric primitives of interest. Starting with the work of Abdalla et al. [ABN10] and Farshim et al. [FLPQ13] (which studied the notion of (key-)robustness in the public-key setting) we develop appropriate notions for symmetric encryption and MACs. To the best of our knowledge, this is the first attempt in this direction (we note that [Moh10] considers robustness and anonymity of hybrid encryption, but not for symmetric encryption directly). As briefly mentioned above, our

notion also formalizes the non-existence of "unexpected collisions" in a cryptosystem over distinct keys, even when inputs (including keys) are maliciously generated.

In Section 4.2 we consider both notions: (1) where the adversary has control over the keys and (2) where the keys are generated honestly. The strongest concept that we formulate is called *complete robustness* and allows an adversary to create the keys used in the system. We show that whether the adversary is in control of the keys or not makes a significant difference, by giving separations between the notions. While previous work in the public-key setting also had to deal with adversarially generated keys that were also *invalid*, this is *not* an issue in our setting, since in the symmetric world keys are often bit-strings of some pre-specified length and can be easily checked for validity. By focusing on correctly formed keys, we can show equivalence between *complete robustness* and a syntactically simpler notion, which we call *full robustness*.

By giving appropriate separating examples, we show that AE security and strong unforgeability do not provide full robustness. Before building fully robust schemes, we first characterize the level of robustness that *is* enjoyed by AE-secure encryption and strongly unforgeable MACs (Section 4.3). For MACs, we prove that as long as the two keys are honestly generated and remain outside the view of the adversary, the scheme is robust in the presence of tag-generation and verification routines. Interestingly, AE-secure encryption schemes achieve a higher level of robustness where both keys are honestly generated, but one is provided to the adversary. Intuitively, this gap arises from the fact that the adversary against the MAC can still choose a message with respect to which a common tag should verify under two distinct keys, but in the encryption setting such an adversary is bound to ciphertexts that are random and outside its control. Unfortunately, these weaker notions of security provide guarantees only if the keys are honestly and independently generated. Therefore, no assurances are provided in applications where the adversary completely controls the keys in the system (like the OT in Example 2), where encryption is performed using related keys, or when the scheme is used to encrypt key-dependent messages (KDM). Full robustness, on the other hand, would be sufficient in such settings.

We then show that full robustness composes well: any fully robust symmetric encryption when combined with a fully robust MAC results in a fully robust AE scheme. Analogous composition results also hold for MAC-then-Encrypt and Encrypt-and-MAC. In these transformations, however, the length of the key doubles (since independent keys are used for encryption and MAC), while in practical AE schemes it is desirable to use a single key for both tasks. Using a *single key* for both the encryption and MAC components not only reduces storage, but it also increases security by solely relying on the robustness of *either* of its components. We emphasize, however, that AE security of the generically composed scheme with key reuse, although provable for some schemes, does not always hold. We show that this can be avoided by modifying the Encrypt-then-MAC transform also to *authenticate the encryption key*. As long as the MAC component is both pseudorandom and collision-resistant, we show this transform gives a robust and AE-secure scheme. Simultaneous pseudorandomness and collision-resistance is an expected property from standard hash functions (and is met by the random oracle); this provides the most practical route to build robust encryption schemes generically. We caution, however, that not all MACs would satisfy this requirement. In particular, we point out that CBC-MAC fails to be fully robust, even when one of two honestly generated keys is in adversary's view.

We then ask if feasibility results for robustness in the public-key setting can be translated to the symmetric setting; this turns out *not* to be the case. The main reason for this is

that in the asymmetric setting the public key can be used as a mechanism to *commit* to its associated secret key. In the symmetric case, on the other hand, there is no such public information. It might be tempting to think that one can just commit to the secret key and append it to the ciphertext. Unfortunately, this approach cannot be proven secure due to a circular *key-dependency* between the encryption and the commitment components. To give a provably secure construction, we construct appropriate commitments that can be used in this setting. This requires a *right-injective* PRG (described earlier in Chapter 3) that can be in turn based on one-way permutations; this result relies on the one-time security of the MAC and its collision-resistance, which once again we base on right-injective PRGs.

Finally, we investigate the meaning of robustness for signature schemes, by transposing the definitions of SROB and CROB from MACs to a public-key setting (Section 4.5). We show that under correct key-generation, any EUF-CMA-secure signature scheme reaches strong robustness, but point out about the existence of digital signatures that fail to be CROB. In the spirit of the transforms we proposed for MACs, we give a simple generic transform for signature schemes.

4.2 Robust MACs and Authenticated Encryption

4.2.1 Definitions

$\underline{\mathrm{AE}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)}$:	Proc. $\textsc{Enc}(M)$:	Proc. $\textsc{Dec}(C)$:
$b \leftarrow\!\!\$\ \{0,1\}$; $\mathrm{L} \leftarrow \emptyset$	$C \leftarrow\!\!\$\ \mathsf{Enc}(K, M)$	if $C \in \mathrm{L}$ then return $\perp$
$K \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	if $b = 0$ then $C \leftarrow\!\!\$\ \{0,1\}^{\lvert C\rvert}$	$M \leftarrow \mathsf{Dec}(K, C)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(1^\lambda)$	$\mathrm{L} \leftarrow \mathrm{L} \cup \{C\}$	if $b = 0$ then $M \leftarrow \perp$
return $b' = b$	return C	return M
$\underline{\mathrm{SUF}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$, $\boxed{\$\mathrm{UF}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$:	Proc. $\textsc{Tag}(M)$:	Proc. $\textsc{Ver}(M, T)$:
$b \leftarrow\!\!\$\ \{0,1\}$; $\mathrm{L} \leftarrow \emptyset$	$T \leftarrow\!\!\$\ \mathsf{Tag}(K, M)$	if $(M, T) \in \mathrm{L}$ then return $\perp$
$K \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	$\boxed{\text{if } b = 0 \text{ then } T \leftarrow\!\!\$\ \{0,1\}^{\lvert T\rvert}}$	$d \leftarrow \mathsf{Ver}(K, M, T)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Tag},\textsc{Ver}}(1^\lambda)$	$\mathrm{L} \leftarrow \mathrm{L} \cup \{(M, T)\}$	if $b = 0$ then $d \leftarrow 0$
return $b' = b$	return T	return d

Figure 4.1: Games defining the security of authenticated encryption (top), and $\boxed{\text{pseudorandom}}$ and strongly unforgeable message authentication codes (bottom). The AE and \$UF notions entail strong notions of key anonymity for each primitive. IND\$ security is a weakening of AE security where the adversary is not allowed to call the decryption oracle. The standard strong unforgeability game omits the boxed statement from the Tag procedure.

Authenticated encryption. An authenticated encryption scheme AE is a triple of algorithms $\mathsf{AE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ such that: (1) $\mathsf{KGen}(1^\lambda)$ is the randomized key-generation algorithm that on input the security parameter 1^λ outputs a key K; (2) $\mathsf{Enc}(K, M; R)$ is the randomized encryption algorithm that on input a key K, a plaintext M and possibly random coins R outputs a ciphertext C; (3) $\mathsf{Dec}(K, C)$ is the deterministic decryption algorithm that on input a key K and a ciphertext C, outputs a plaintext M or the special error symbol

$\perp$. We call a scheme AE (perfectly) correct (for message space $\{0,1\}^*$) if for all $\lambda \in \mathbb{N}$, all $K \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, all $M \in \{0,1\}^*$ and all $C \leftarrow\!\!\$\ \mathsf{Enc}(K, M)$ we have that $\mathsf{Dec}(K, C) = M$. We define the advantage of an adversary $\mathcal{A}$ against AE as

$$\mathsf{Adv}^{\mathrm{ae}}_{\mathsf{AE},\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr\left[\mathrm{AE}^{\mathcal{A}}_{\mathsf{AE}}(\lambda) = 1\right] - 1 \ ,$$

where game $\mathrm{AE}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)$ is shown in Figure 4.1 (top). An AE scheme is secure if the above advantage function in negligible for every PPT adversary $\mathcal{A}$. This is the standard definition of security for AE schemes [RS06; HK07]. An alternative security definition would come with a challenge oracle that on input two messages (M_0, M_1) of the same length, returns an encryption of M_b. This definition is weaker than AE security, as the latter already implies a strong form of *anonymity* due to the pseudorandomness of ciphertexts, whereas this is not necessarily the case for the left-right-based definition.[2]

Message authentication codes. A message authentication code (MAC) is a triple of algorithms $\mathsf{MAC} := (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Ver})$ defined as follows: (1) $\mathsf{KGen}(1^\lambda)$ is the randomized key generation algorithm that on input the security parameter 1^λ outputs a key K; (2) $\mathsf{Tag}(K, M; R)$ is the randomized tagging algorithm that on input a key K, a plaintext M and possibly random coins R, outputs a tag T; (3) $\mathsf{Ver}(K, M, T)$ is the deterministic verification algorithm that on input a key K, a plaintext M and a tag T, outputs a bit. We call a MAC scheme (perfectly) correct (for message space $\{0,1\}^*$) if for all $\lambda \in \mathbb{N}$, all $K \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, all $M \in \{0,1\}^*$ and all $T \leftarrow\!\!\$\ \mathsf{Tag}(K, M)$, the verification is successful: $\mathsf{Ver}(K, M, T) = 1$. We define the advantage of an adversary $\mathcal{A}$ against a MAC as

$$\mathsf{Adv}^{\$\mathrm{uf}}_{\mathsf{MAC},\mathcal{A}}(\lambda) \coloneqq 2 \cdot \Pr\left[\$\mathrm{UF}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda) = 1\right] - 1 \ ,$$

where game $\$\mathrm{UF}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)$ is shown in Figure 4.1. This game strengthens the standard strong unforgeability for MACs, which is shown in the same figure omitting the boxed statement, in a number of aspects. First, Ver outputs the error symbol only when the pair (M, T) was generated via the Tag procedure and therefore pseudorandom MACs are also strongly unforgeable. Second, the definition implies the tags are pseudorandom and hence they fully hide the messages and the *keys* that were used to generate them. Stated differently, pseudorandom MACs are both *confidential* and *anonymous* in the sense that they hide both the message and the key that is used to generate a tag. Finally, since Tag does not repeat T for repeated messages, it implies a notion of *unlinkability* [BFLS10].

Feasibility of pseudorandom MACs. Given a PRF, consider a scheme MAC whose key-generation algorithm is identical to that of the PRF and whose tag-generation and verification algorithms operate as

$$\mathsf{Tag}(K, M; R) \coloneqq R || \mathsf{PRF}(K, M||R), \quad \mathsf{Ver}(K, M, (R||T)) \coloneqq (T \stackrel{?}{=} \mathsf{PRF}(K, M||R)) \ .$$

We call such message authentication codes *randomized* MAC*s*. It is straightforward to prove that this MAC satisfies our strong security notion for MACs given above.

Encrypt-then-MAC. Recall that in the Encrypt-then-MAC paradigm, one first encrypts a message M and finally authenticates the resulting ciphertext using a MAC. If the underlying encryption AE in this transform is AE-secure without access to decryption oracle (a.k.a. IND$ secure) and the MAC used is pseudorandom, the encryption scheme is AE secure [BN08].

[2] The left-right-based definition can be modified to imply a left-right notion of key anonymity. The resulting game, however, is both more cumbersome to work with and weaker than standard AE security.

$\underline{\text{CROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$\varepsilon \leftarrow\!\!\$\ \mathcal{A}^{\text{ENC},\text{DEC}}(1^\lambda)$
for $(K_1, M_1, C_1), (K_2, M_2, C_2) \in \mathrm{L}$ do
  if $(C_1 = C_2 \neq \bot) \wedge (K_1 \neq K_2) \wedge (M_1 \neq \bot \wedge M_2 \neq \bot)$
    return 1
return 0

$\underline{\text{Proc. ENC}(K, M, R)}$:
$C \leftarrow \mathsf{Enc}(K, M; R)$
$\mathrm{L} \leftarrow \mathrm{L} \cup (K, M, C)$

$\underline{\text{Proc. DEC}(K, C)}$:
$M \leftarrow \mathsf{Dec}(K, C)$
$\mathrm{L} \leftarrow \mathrm{L} \cup (K, M, C)$

$\underline{\text{CROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$\varepsilon \leftarrow\!\!\$\ \mathcal{A}^{\text{TAG},\text{VER}}(1^\lambda)$
for $(K_1, M_1, T_1), (K_2, M_2, T_2) \in \mathrm{L}$ do
  if $(T_1 = T_2 \neq \bot) \wedge (K_1 \neq K_2)$ then
    return 1
return 0

$\underline{\text{Proc. TAG}(K, M, R)}$:
$T \leftarrow \mathsf{Tag}(K, M; R)$
$\mathrm{L} \leftarrow \mathrm{L} \cup (K, M, T)$

$\underline{\text{Proc. VER}(K, M, T)}$:
$b \leftarrow \mathsf{Ver}(K, M, T)$
if $b = 1$ then $\mathrm{L} \leftarrow \mathrm{L} \cup (K, M, T)$

Figure 4.2: Complete robustness for symmetric encryption (top) and MAC (bottom).

ROBUSTNESS FOR SYMMETRIC-KEY PRIMITIVES. Informally, in a robust scheme, no unexpected collisions in the input/output behaviour of the system exist. For instance, in the case of encryption, no adversary should be able to compute a ciphertext that decrypts correctly under two distinct keys. This notion was first formulated in the asymmetric setting [ABN10; FLPQ13] and we adapt it to authenticated encryption and MACs in the current chapter.

The work of [FLPQ13] refines and strengthens the original definitions of robustness [ABN10]. The central security notion introduced in [FLPQ13] is *complete robustness* (CROB), a notion that contains three sub-notions of full robustness (FROB), key-less robustness (KROB) and mixed robustness (XROB). These, roughly speaking, correspond to three possible ways of finding a colliding ciphertext using either the encryption or decryption algorithms of the scheme. That is, for some $K_1, K_2, M_1, M_2, R_1, R_2, C_1, C_2$ at least one of the checks

$$\mathsf{Enc}(K_1, M_1; R_1) = \mathsf{Enc}(K_2, M_2; R_2), \quad \text{or} \quad \mathsf{Dec}(K_1, C_1) = \mathsf{Dec}(K_2, C_1), \quad \text{or}$$

$$\mathsf{Dec}(K_2, \mathsf{Enc}(K_1, M_1; R_1)) = M_2,$$

pass when $K_1 \neq K_2$. The last condition can be made stronger: one expects that $\mathsf{Enc}(K_1, M_1; R_1)$ would decrypt to $\bot$ under an unrelated key K_2. The middle check can also be made stronger by only checking that the outputs are both valid. We formalize the resulting notions next.

We define the advantage of an adversary $\mathcal{A}$ in the CROB games against an encryption scheme AE as

$$\mathsf{Adv}^{\text{crob}}_{\mathsf{AE},\mathcal{A}}(\lambda) := \Pr\left[\text{CROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda) = 1\right] ,$$

where game $\text{CROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)$, is shown in Figure 4.2 (top). Similarly, for a message authentication code MAC we define

$$\mathsf{Adv}^{\text{crob}}_{\mathsf{MAC},\mathcal{A}}(\lambda) := \Pr\left[\text{CROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda) = 1\right] ,$$

where $\text{CROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)$ is shown in Figure 4.2 (bottom).

Farshim et al. [FLPQ13] give pair-wise separations among the three sub-notions mentioned above, showing they are all incomparable and hence should be (implicitly) included in the

CROB notion. Some of these separations use *invalid* keys, as key pairs in the public-key setting cannot be necessarily checked for validity. This issue disappears in our setting as the key space is $\{0,1\}^k$, a set which is trivially checkable for validity. This fact simplifies relations among notions (which we study in detail in Section 4.2.3). Analogues of the FROB notion [FLPQ13] for AE and MAC turn out to be equivalent to our strongest notions above. We formalize FROB in Figure 4.3 (left) for AE and (right) for MACs, and summarize this discussion under Theorem 4.1. Previously (Chapter 3, Figure 3.9) we have already defined the robustness experiment for PRFs with advantage function

$$\mathsf{Adv}^{\mathrm{frob}}_{\mathsf{PRF},\mathcal{A}}(\lambda) := \Pr\left[\mathrm{FROB}^{\mathcal{A}}_{\mathsf{PRF}}(\lambda) = 1\right] .$$

As we shall see, from a foundational perspective, robust PRFs underlie feasibility of robustness for many symmetric primitives.

$\underline{\mathrm{FROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)}$:	$\underline{\mathrm{FROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$:
$(C, K_1, K_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$	$(T, K_1, M_1, K_2, M_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
if $K_1 = K_2$ return 0	if $K_1 = K_2$ return 0
$M_1 \leftarrow \mathsf{Dec}(K_1, C)$	$d_1 \leftarrow \mathsf{Ver}(K_1, M_1, T)$
$M_2 \leftarrow \mathsf{Dec}(K_2, C)$	$d_2 \leftarrow \mathsf{Ver}(K_2, M_2, T)$
return $(M_1 \neq \perp \wedge M_2 \neq \perp)$	return $(d_1 = d_2 = 1)$

Figure 4.3: Games defining full robustness for a symmetric encryption scheme AE (left), a message authentication code MAC (right).

Collision resistance. Complete robustness is strengthened to *unkeyed* collision resistance when the case $K_1 = K_2$ is not ruled out. For MACs, (unkeyed) collision-resistance states that it should be hard to come up with $(K_1, M_1, R_1) \neq (K_2, M_2, R_1)$ such that $\mathsf{Tag}(K_1, M_1; R_1) = \mathsf{Tag}(K_2, M_2; R_1)$. Unkeyed collision resistance of PRFs requires that $\mathsf{PRF}(K_1, M_1) \neq \mathsf{PRF}(K_2, M_2)$ for $(K_1, M_1) \neq (K_2, M_2)$ (Section 3.4). The standard notion of keyed collision resistance, on the other hand, imposes that keys are equal, $K_1 = K_2$, and are honestly generated. (Note that unforgeable MACs and pseudorandom PRFs are always keyed collision-resistant.)

4.2.2 Implications

Theorem 4.1 (Robustness with key validity)**.** *Let* AE *be a perfectly correct symmetric encryption scheme that checks keys for validity during encryption. Then* AE *is* CROB *secure if and only if it is* FROB *secure. Similarly, a perfectly correct message authentication code* MAC *whose tag-generation algorithm checks keys for validity is* CROB *secure if and only if it is* FROB *secure.*

Theorem 4.1. The proof is simple, and we give an example for one case. Suppose that adversary $\mathcal{A}$ wins the CROB game by finding a collision between the outputs of encryption. In other words, $\mathcal{A}$ finds $(K_1, M_1, R_1, K_2, M_2, R_2)$ such that

$$\mathsf{Enc}(K_1, M_1; R_1) = \mathsf{Enc}(K_2, M_2; R_2) .$$

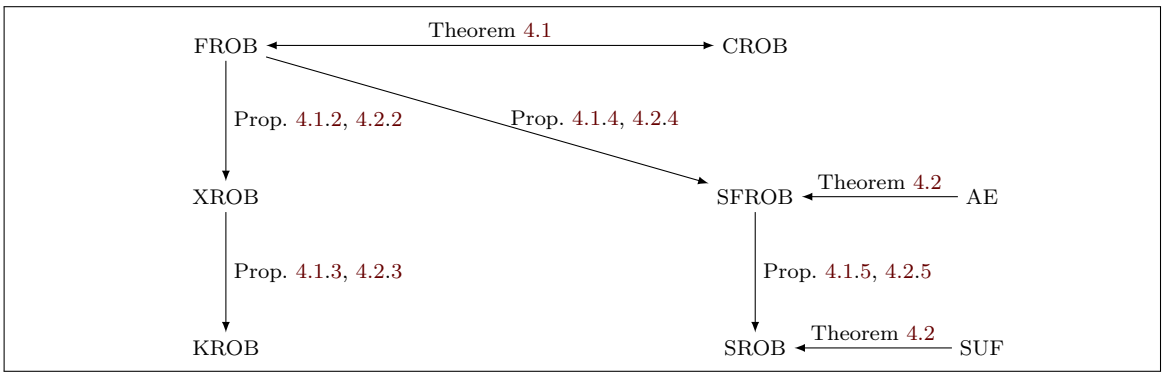

Figure 4.4: Relations among notions of robustness (with key validity) for AE and MAC schemes. XROB and KROB are formalized in Section 4.2.3, and SFROB and SROB are formalized in Section 4.3.

This means that K_1 and K_2 are valid. Now consider a FROB adversary that computes $C := \mathsf{Enc}(K_1, M_1; R_1)$ and outputs (C, K_1, K_2). By the perfect correctness of the scheme for *valid keys*, it must be the case that $\mathsf{Dec}(K_1, C) = M_1$ and $\mathsf{Dec}(K_2, C) = M_2$, which wins the FROB game.

Other cases are dealt similarly, by either computing a colliding ciphertexts using Enc or a colliding tag using Tag. We provide the details in the following part. □

4.2.3 Further Relations among Notions of Robustness

For completeness and comparison with prior work, we introduce symmetric analogues of mixed-robustness (XROB) and keyless-robustness (KROB) for AE schemes in Figure 4.5 below. This follows the definitions of [FLPQ13] in the context of public-key encryption.

$\underline{\mathrm{XROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)}$:
$(M_1, K_1, R_1, C_2, K_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1)$
$M_2 \leftarrow \mathsf{Dec}(K_2, C_2)$
if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$
  $C_1 = C_2 \neq \bot$ return 1
return 0

$\underline{\mathrm{KROB}^{\mathcal{A}}_{\mathsf{AE}}(\lambda)}$:
$(M_1, K_1, R_1, M_2, K_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1)$
$C_2 \leftarrow \mathsf{Enc}(K_2, M_2; R_2)$
if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$
  $C_1 = C_2 \neq \bot$ return 1
return 0

Figure 4.5: Mixed robustness (XROB) andkey-less robustness (KROB) for AE.

We study relations among notions of robustness for AE schemes below.

Proposition 4.1. *Let* AE *be an encryption scheme.*

1. AE *is* FROB *secure if and only if it is* CROB *secure.*
2. *If* AE *is* FROB *secure, then it is also* XROB *secure.*
3. *If* AE *is* XROB *secure, then it is also* KROB *secure.*
4. *If* AE *is* FROB *secure, then it is also* SFROB *secure.*

5. If AE *is* SFROB *secure, then it is also* SROB *secure.*

Proof. (1) FROB $\iff$ CROB**.** ("$\Leftarrow$") Assume the existence of an adversary that wins the FROB game. Then this adversary also wins the CROB game by querying the FROB winning tuples to the Dec oracle. ("$\Rightarrow$") First, from (2) and (3) we have that a FROB scheme is also KROB and XROB. We then note that a pair of winning tuples for the CROB game can arise in one of three possible ways: (1) Both tuples were added to the list through decryption queries. This directly translates into a winning output for a FROB adversary; (2) Both tuples were added to the list through encryption queries. This translates into a winning output for a KROB adversary; (3) One tuple was added to the list through an encryption query and the other through a decryption query. This translates into a winning output for an XROB adversary.

(2) FROB $\implies$ XROB**.** We proceed as in the previous case. We build an adversary $\mathcal{B}$ that wins the FROB game in Figure 4.6. $\mathcal{B}$ runs $\mathcal{A}$ to obtain an XROB winning tuple $(M_1, K_1, R_1, C_2, K_2)$ that fulfils the XROB constraints: $C_1 = \mathsf{Enc}(K_1, M_1; R_1) = C_2 \wedge \mathsf{Dec}(K_2, C_2) \neq \perp$. Then $\mathcal{B}$ computes $C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1)$ and uses the tuple (C_1, K_1, K_2) to win the FROB game: both tuples $\mathsf{Dec}(K_1, C_1)$ and $\mathsf{Dec}(K_2, C_2)$ will return $\neq \perp$, given that C is a valid ciphertext. Therefore $\mathsf{Adv}^{\mathrm{frob}}_{\mathsf{AE},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathrm{xrob}}_{\mathsf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(M_1, K_1, R_1, C_2, K_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
2. $C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1)$
3. return (C_1, K_1, K_2)

Figure 4.6: FROB $\Rightarrow$ XROB.

(3) XROB $\implies$ KROB**.** The intuition behind the proof is that an adversary breaking KROB can be used to construct an XROB winning tuple simply by encrypting part of the output obtained from the KROB adversary. The reduction is shown in Figure 4.7. Let $\mathcal{A}$ be an adversary having a non-negligible advantage against the KROB game. We build an adversary $\mathcal{B}$ that wins the XROB game as follows: $\mathcal{B}$ begins by running $\mathcal{A}$ to obtain a KROB winning tuple $(M_1, K_1, R_1, M_2, K_2, R_2)$ that fulfils the KROB constraint: $C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1) \wedge C_2 \leftarrow \mathsf{Enc}(K_2, M_2; R_2) \wedge C_1 = C_2$. Next, $\mathcal{B}$ computes $C_2 \leftarrow \mathsf{Enc}(K_2, M_2; R_2)$ and creates the tuple $(M_1, K_1, R_1, C_2, K_2)$ to win the XROB game; we state that $C_1 \leftarrow \mathsf{Enc}(K_1, M_1; R_1) \neq \perp$ because it is part of a KROB tuple while $\mathsf{Dec}(K_2, C_2) \neq \perp$ returns a valid message with non-negligible probability. We conclude that $\mathsf{Adv}^{\mathrm{xrob}}_{\mathsf{AE},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathrm{krob}}_{\mathsf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(M_1, K_1, R_1, M_2, K_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
2. $C_2 \leftarrow \mathsf{Enc}(K_2, M_2; R_2)$
3. return $(M_1, K_1, R_1, C_2, K_2)$

Figure 4.7: XROB $\Rightarrow$ KROB.

(4) FROB $\implies$ SFROB**.** As in the previous cases, we build an adversary $\mathcal{B}$ that wins the FROB game in Figure 4.8. $\mathcal{B}$ samples K_1, K_2 uniformly at random and runs $\mathcal{A}$ and answers its oracle queries using the keys. When $\mathcal{A}$ returns C; then, $\mathcal{B}$ constructs an FROB winning tuple (C, K_1, K_2) that fulfils the constraints: $M_1 \leftarrow \mathsf{Dec}(K_1, C) \wedge M_2 \leftarrow \mathsf{Dec}(K_2, C) \wedge M_1 \neq \perp \wedge M_2 \neq \perp$. $\mathcal{B}$ simply returns (C, K_1, K_2) to win the FROB game. Therefore $\mathsf{Adv}^{\mathrm{frob}}_{\mathsf{AE},\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathrm{sfrob}}_{\mathsf{AE},\mathcal{A}}(\lambda)$.

Algorithm $\mathcal{B}(1^\lambda)$:
1. $(K_1, K_2) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
2. $C \leftarrow\!\!\$\ \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(1^\lambda, K_2)$
3. return (C_1, K_1, K_2)

Figure 4.8: FROB $\Rightarrow$ SFROB.

(5) SFROB $\implies$ SROB**.** This follows from a trivial reduction as the games are identical except that an SROB adversary does not get to see K_2. □

We define mixed-robustness (XROB) and keyless-robustness (KROB) for MACs in Figure 4.9 below.

$\underline{\mathrm{XROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$:
$(M_1, K_1, R_1, M_2, K_2, T_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$T_1 \leftarrow \mathsf{Tag}(K_1, M_1; R_1)$
$b_2 \leftarrow \mathsf{Ver}(K_2, M_2, T_2)$
if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$
$b_2 = 1 \wedge T_1 = T_2 \neq \bot$ return 1
return 0

$\underline{\mathrm{KROB}^{\mathcal{A}}_{\mathsf{MAC}}(\lambda)}$:
$(M_1, K_1, R_1, M_2, K_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$T_1 \leftarrow \mathsf{Tag}(K_1, M_1; R_1)$
$T_2 \leftarrow \mathsf{Tag}(K_2, M_2; R_2)$
if $M_1 \neq \bot \wedge M_2 \neq \bot \wedge K_1 \neq K_2 \wedge$
$T_1 = T_2 \neq \bot$ return 1
return 0

Figure 4.9: Mixed robustness (XROB) and key-less robustness (KROB) for MAC.

Proposition 4.2. *Let* MAC *be a message authentication code.*

1. *A* MAC *is* FROB *secure if an only if it is* CROB *secure.*
2. *If* MAC *is* FROB *secure, then it is also* XROB *secure.*
3. *If* MAC *is* XROB *secure, then it is also* KROB *secure.*
4. *If* MAC *is* FROB *secure, then it is also* SFROB *secure.*
5. *If* MAC *is* SFROB *secure, then it is also* SROB *secure.*

The proofs are omitted as they are virtually identical to those of Proposition 4.1.

Throughout the chapter, we assume that keys are checkable for validity and that they are indeed checked for validity in all algorithms. Hence, we will only use FROB security to establish CROB security in the subsequent sections. We limit our study to schemes that have *perfect* correctness (as defined under syntax). Correctness with all but negligible probability would allow for artificial attacks and separations. As an example, consider an encryption scheme that, when invoked with a special random tape computes the identity function – this is allowed since the probability of hitting that random tape is negligible and at the same time gives an easy way to break robustness.

4.3 Robustness, AE Security, and Unforgeability

We show that standard AE-secure encryption schemes offer a basic level of resilience against incorrect usage of keys. The level of robustness offered corresponds to a setting where the adversary does not get to choose any keys. Instead, two keys are honestly generated and the adversary is given oracle access to encryption and decryption algorithms under *both* keys. The notion for MACs is similar where oracle access to tag-generation and verification algorithms under honestly generated keys are provided to the adversary. This notion, which we call *strong robustness* (SROB) is shown in Figure 4.10 (without boxed variables). This nomenclature follows the original notion of strong robustness by Abdalla et al. [ABN10]. We also define *semi-full robustness* (SFROB) as one where the adversary gets to see one of the keys (as shown in Figure 4.10 with boxed variables).

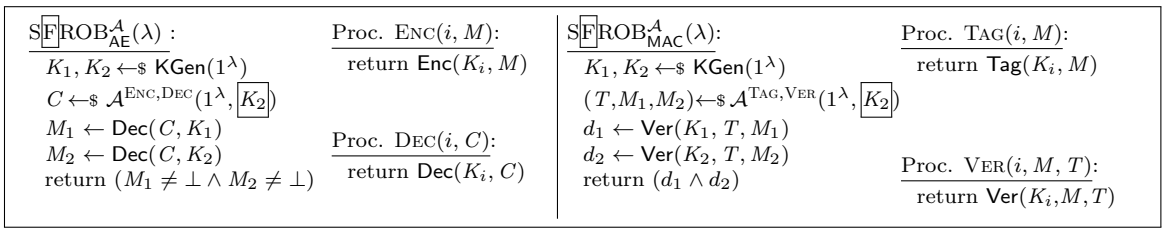

Figure 4.10: Games defining strong robustness (SROB) and semi-full robustness (SFROB) for symmetric encryption (left) and MACs (right). The boxed statements are included in the boxed games.

Theorem 4.2. *Any authentication scheme* AE *that is* AE*-secure is also* SFROB*-secure. Any strongly unforgeable (and in particular pseudorandom) scheme* MAC*, on the other hand, is (only)* SROB*-secure. More precisely, for any adversary* $\mathcal{A}$ *against the* SFROB *security of the AE scheme, there is an adversary* $\mathcal{B}$ *against the* AE *security of the scheme such that*

$$\mathsf{Adv}^{\text{sfrob}}_{\mathsf{AE},\mathcal{A}}(\lambda) \leq 3 \cdot \mathsf{Adv}^{\text{ae}}_{\mathsf{AE},\mathcal{B}}(\lambda) \ .$$

Moreover, for any adversary $\mathcal{A}$ *against the* SROB*-security of the MAC there is an adversary* $\mathcal{B}$ *against the* SUF*-security of the scheme such that*

$$\mathsf{Adv}^{\text{srob}}_{\mathsf{MAC},\mathcal{A}}(\lambda) \leq 3 \cdot \mathsf{Adv}^{\text{suf}}_{\mathsf{AE},\mathcal{B}}(\lambda) \ .$$

Furthermore, there exist a pseudorandom MAC that is not SFROB*-secure.*

Theorem 4.2. First, we prove the implication from AE security to SFROB. Let Game_0 be the SFROB game. We assume without loss of generality that the adversary in Game_0 never queries the $\text{Enc}(2, \cdot)$ and $\text{Dec}(2, \cdot)$ oracles as it has access to K_2, and that it never queries an output of $\text{Enc}(1, \cdot)$ to $\text{Dec}(1, \cdot)$ as it already knows the answer.

In Game_1 we modify the winning condition of Game_0 as follows. When the adversary returns a ciphertext C, instead of checking that $\mathsf{Dec}(C, K_1) \neq \bot$ and $\mathsf{Dec}(C, K_2) \neq \bot$, the game checks if C was one of the ciphertexts that was returned from the $\text{Enc}(1, \cdot)$ oracle and that $\mathsf{Dec}(C, K_2) \neq \bot$. The games Game_0 and Game_1 are identical unless $\mathcal{A}$ outputs a ciphertext C that was not obtained from the $\text{Enc}(1, \cdot)$ oracle, but decrypts correctly (call this event E). We bound the probability of E via the AE as follows. For any distinguishing $\mathcal{A}$, we define an algorithm $\mathcal{B}$ that picks a random key K_2, runs $\mathcal{A}(K_2)$, and answers its queries using its own equivalent pair of oracles. When $\mathcal{A}$ terminates with C, algorithm $\mathcal{B}$ queries C to its decryption oracle to get M_1 and also computes $M_2 \leftarrow \mathsf{Dec}(C, K_2)$. It returns $(M_1 \neq \bot \wedge M_2 \neq \bot)$. If $\mathcal{B}$'s decryption oracle is fake and implements $\bot$, algorithm $\mathcal{B}$ will always return 0. If $\mathcal{B}$'s decryption oracle is real, algorithm $\mathcal{B}$ runs $\mathcal{A}$ according to the environment of Game_0 and Game_1 and will output 1 whenever E happens. Hence $\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}] \leq \Pr[E] = \mathsf{Adv}^{\text{ae}}_{\mathsf{AE},\mathcal{B}}(\lambda)$.

In Game_2 we replace $\text{Enc}(1, \cdot)$ and $\text{Dec}(1, \cdot)$ with the \$ and $\bot$ procedures respectively. (As in Game_1 we still use the list of ciphertexts and K_2 for the winning condition). The distance between Game_1 and Game_2 can be bounded via the AE game as follows. Consider an AE adversary $\mathcal{B}$ that generates an independent key K_2 and runs a distinguishing adversary $\mathcal{A}(K_2)$. Algorithm $\mathcal{B}$ answers $\mathcal{A}$'s oracle queries using the oracles provided to it. When $\mathcal{A}$ terminates with a ciphertext C, algorithm $\mathcal{B}$ performs the winning check and outputs its

result. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ with respect to the real or replaced procedures according to the real or fake procedures that it gets. The output of $\mathcal{B}$ is identical to that of $\mathcal{A}$ in the two games. Hence $\Pr[\text{Game}_1^{\mathcal{A}}] - \Pr[\text{Game}_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\text{ae}}_{\mathsf{AE},\mathcal{B}}(\lambda)$.

In Game_2, the adversary has essentially no control over C, and we show its advantage is small. For this, we will rely on the AE game once more (but now implicitly concerning the second key). Game_2 can only be won if $\mathsf{Dec}(K_2, C) \neq \perp$ for at least one of the q distinct random strings C obtained from the \$ oracle. Consider an AE adversary $\mathcal{B}$ that generates q such random C and queries them to its DEC oracle and outputs 1 if and only if one of the answers is non-$\perp$. Adversary $\mathcal{B}$ always outputs 0 when the oracle implements $\perp$. On the other hand, when the oracle implements the real decryption routine, the probability of $\mathcal{B}$ outputting 1 is exactly the probability that $\mathsf{Dec}(K_2, C) = \perp$ for one of the random C and key K_2. This means $\Pr[\text{Game}_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\text{ae}}_{\mathsf{AE},\mathcal{B}}(\lambda)$. The first part of the theorem follows from that last (in)equalities.

We now prove the second part of the Theorem 4.2. We first note that via a simple hybrid argument, unforgeability regarding the two keys reduces to unforgeability with respect to a single key with loss 2 in advantage. We also assume, without loss of generality, that an adversary in $\text{Game}_0 :=$ SROB does not query VER on any (i, M, T) where T is an output of $\text{TAG}(i, M)$; the answer is always 1 for such queries.

In Game_1 we replace the $\text{VER}(1, \cdot, \cdot)$ and $\text{VER}(2, \cdot, \cdot)$ procedures with the $\perp$ procedure. We also replace the computation of $\mathsf{Ver}(K_i, T, M)$ for $i = 1, 2$ in the winning condition with 0 unless T was output by *both* $\text{TAG}(1, \cdot)$ and $\text{TAG}(2, \cdot)$ procedures. Hence Game_0 and Game_1 are identical unless $\mathcal{A}$ outputs a tag T that was not output by both tag-generation oracles and yet verifies under both keys. Call this event E. The probability of event E can be bounded via the (single-key) SUF game as follows. Algorithm $\mathcal{B}$ generates a key K_2. It uses its own oracles and K_2 to simulate the oracles for $\mathcal{A}$. When $\mathcal{A}$ terminates with a tag (T, M_1, M_2), algorithm $\mathcal{B}$ queries (T, M_1) to its verification oracle and returns 1 iff the result was not 0. Algorithm $\mathcal{B}$ will always output 0 when the oracle is 0 (i.e., when it is fake). If its oracles are real, $\mathcal{B}$ runs $\mathcal{A}$ according to the environments of Game_0 and Game_1, and whenever E happens, it returns 1. Hence, $\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}] \leq \Pr[E] = \mathsf{Adv}^{\text{suf}}_{\mathsf{MAC},\mathcal{B}}(\lambda)$.

The advantage of any adversary $\mathcal{A}$ in Game_1 can be bounded, once again, by the two-key SUF game as follows. Consider any adversary against the two-key SUF game. Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and answers its oracle queries using its own oracles. When $\mathcal{A}$ terminates with a tag (T, M_1, M_2), algorithm $\mathcal{B}$ checks for which i this tag was not obtained from $\text{TAG}(i, \cdot)$ (if both, it chooses either i). Algorithm $\mathcal{B}$ then queries $\text{VER}(i, T, M_i)$ and returns 1 if and only if the result is not 0. Note that $\mathcal{B}$ never outputs 1 when its oracles are fake. However, when its oracles are real $\mathcal{B}$ runs $\mathcal{A}$ according to the rules of Game_1, and it returns 1 whenever $\mathcal{A}$ wins. Hence, $\Pr[\text{Game}_2^{\mathcal{A}}] \leq 2 \cdot \mathsf{Adv}^{\text{suf}}_{\mathsf{MAC},\mathcal{B}}(\lambda)$. The second part of the theorem follows.

Interestingly, MAC security (including pseudorandomness) does not imply SFROB security for MACs. (And the above theorem is, in a sense, "sharp"). Indeed, given a pseudorandom MAC consider a modified scheme whose verification procedure on input $M = K$ and *any* tag always passes. This MAC can be still shown to be pseudorandom (without access to K), but fails to be SFROB as any tag T obtained under K_1 for, say, message 0 would also be valid with respect to K_2 if message $M_2 := K_2$. Note, however, that since any AE scheme is a pseudorandom MAC, the result for AE schemes shows SFROB-secure MACs can be built via authenticated encryption. □

In the above proof, we showed that for MACs, SROB is strictly weaker than SFROB, and hence it is also weaker than CROB. We next prove that SFROB is weaker than CROB for AE schemes. We show a stronger result that not all AE schemes, even those obtained via Encrypt-then-MAC, are CROB.

Proposition 4.3. *There exist an authenticated encryption scheme obtained via the Encrypt-then-MAC transform that is not* CROB *secure (but* SFROB *secure as shown in Theorem 4.2).*

Proposition 4.3. Consider any symmetric encryption scheme whose decryption algorithm never outputs $\perp$. (A natural example is a scheme whose encryption algorithm evaluates a PRF at a random point and masks the message with the result: $\mathsf{Enc}(K_e, M; R) := R||\mathsf{Eval}(K_e, R) \oplus M$). Then, the AE scheme obtained by applying the EtM transform using such an encryption scheme and *any* MAC (even robust ones) will not be CROB secure. For a random MAC key K_m and random and distinct encryption keys K_{e_1}, K_{e_2} consider an attacker that computes $C \leftarrow\!\!\$\ \mathsf{Enc}(K_{e_1}, 0)$ and $T \leftarrow\!\!\$\ \mathsf{Tag}(K_m, C)$ and outputs $((C||T), (K_{e_1}||K_m), (K_{e_2}||K_m))$. The ciphertext $(C||T)$ will decrypt to a valid message under the distinct keys $(K_{e_1}||K_m)$ and $(K_{e_2}||K_m)$ as the tag T is always checked against K_m and the base encryption scheme does not have invalid ciphertexts. □

The attack described above applies against authenticated encryption schemes that follow the EtM transform and use independent keys for the encryption and MAC components. If the same key is used for both the encryption and authentication components (and assuming the AE security of the composed construction), the above attack no longer works. Artificial counterexamples, however, still exist. As before, consider a MAC that verifies whenever $M = K$ irrespectively of its input tag. Such a MAC, when combined with any encryption scheme whose decryption never returns $\perp$ gives rise to a separating example between CROB and SFROB for AE schemes. Here the attacker gets K_2, sets $C := K_2$, computes a tag $T \leftarrow \mathsf{Tag}(K_1, C)$ and outputs $((C||T), K_1, K_2)$. Now the verification of T for C with K_1 always passes. It also passes with respect to K_2 and $K_2 = C$. Since Dec never outputs $\perp$ in the base scheme, C also decrypts under both keys.

CROB INSECURITY OF CBC-MAC. We conclude this section showing that the popular CBC-MAC is not CROB (or even SFROB) secure as the block cipher used in CBC-MAC is invertible. In CBC-MAC, a tag is generated as $C_i = E(K, C_{i-1} \oplus M_i)$, with $C_0 := IV$ for some fixed IV. To attack the (semi-)full robustness of CBC-MAC, for two random keys K_1, K_2 take any plaintext M, generate $T \leftarrow E(K_1, M_1 \oplus IV)$, compute $M_2' \leftarrow D(K_2, T)$, and set $M_2 := M_2' \oplus IV$. Now (T, K_1, M_1, K_2, M_2) constitutes a break against the (semi-)full robustness of CBC-MAC.

4.4 Constructions

We now prove two positive results for obtaining robust encryption through generic composition.

Theorem 4.3 (Robustness for generic composition)**.** *The* AE *schemes obtained through either Encrypt-then-Mac (*EtM*), Encrypt-and-MAC (*EaM*), or MAC-then-Encrypt (*MtE*) (with independent keys) are* CROB *secure as long as their encryption and MAC components are* CROB *secure. Moreover, the* AE *scheme obtained through* EtM*,* EaM *or* MtE *when reusing the same key for encryption and authentication is* CROB *secure as long as* either *the encryption or the MAC component is* CROB *secure.*

Theorem 4.3. We provide the proofs for the three cases separately. **EtM composition.** Suppose a CROB adversary $\mathcal{A}$ outputs $(C||T, K_{e_1}||K_{m_1}, K_{e_2}||K_{m_2})$, a winning tuple for the CROB game against the generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$ there are two possibilities to consider:

Case $K_{e_1} \neq K_{e_2}$: then (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful.

Case $K_{m_1} \neq K_{m_2}$: then $(T, K_{m_1}, C, K_{m_2}, C)$ wins the CROB game for MAC as T would have to be a valid tag with respect to C and two distinct keys.

To sum up, for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, $\mathsf{Adv}^{\mathrm{crob}}_{\mathsf{EtM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{AE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, we can apply *both* branches of the reduction above. This proves the CROB security of the composed scheme assuming CROB security for *either* the AE or MAC component of the scheme and get $\mathsf{Adv}^{\mathrm{crob}}_{\mathsf{EtM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{AE},\mathcal{B}_1}(\lambda)$ and $\mathsf{Adv}^{\mathrm{crob}}_{\mathsf{EtM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda)$.

EaM composition. Suppose a CROB adversary $\mathcal{A}$ outputs $(C||T, K_{e_1}||K_{m_1}, K_{e_2}||K_{m_2})$, a winning tuple for the CROB game against the EaM generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$, as for the EtM transform, if: (1) $K_{e_1} \neq K_{e_2}$, we have that (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful; (2) for the second case we let $M_1 \leftarrow \mathsf{Dec}(K_{e_1}, C)$ and $M_2 \leftarrow \mathsf{Dec}(K_{e_2}, C)$; when $K_{m_1} \neq K_{m_2}$, then $(T, K_{m_1}, M_1, K_{m_2}, M_2)$ wins the CROB game for MAC as T would have to be a valid tag with respect to M_1, M_2 and both keys for $\mathcal{A}$ to be successful. Thus for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$, the advantage of $\mathcal{A}$ is bounded by: $\mathsf{Adv}^{\mathrm{crob}}_{\mathsf{EaM},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{AE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, the same argument as in the previous case applies.

MtE composition. Let a CROB adversary $\mathcal{A}$ output a tuple $(C, K_{e_1}||K_{m_1}, K_{e_2}||K_{m_2})$ winning the CROB game against the MtE generically composed scheme with distinct keys. Since $(K_{e_1}, K_{m_1}) \neq (K_{e_2}, K_{m_2})$, as for the EtM transform, if: (1) $K_{e_1} \neq K_{e_2}$, we have that (C, K_{e_1}, K_{e_2}) wins the CROB game against encryption, as C would have decrypted correctly with respect to both keys for $\mathcal{A}$ to be successful. Thus we assume $K_{e_1} = K_{e_2}$ and let $(M||T) \leftarrow \mathsf{Dec}(K_{e_1}, C)$; (2) when $K_{m_1} \neq K_{m_2}$ then $(T, K_{m_1}, M, K_{m_2}, M)$ wins the CROB game for MAC as T would have to be a valid tag with respect to M and both keys for $\mathcal{A}$ to be successful. (Note that the same tag is obtained after decryption). Therefore for adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ the advantage of $\mathcal{A}$ is bounded in the following way: $\mathsf{Adv}^{\mathrm{crob}}_{\mathsf{MtE},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{AE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{crob}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda)$. When the keys are reused, the same argument as in the first case applies. □

Some CAESAR [Ber14] candidates follow the generic composition paradigm but incorporate various optimizations to reduce computation, bandwidth and keying material. As many of the candidate constructions are reusing keys for the encryption and authentication components, a proof of robustness for either of their components would suffice to show (under Theorem 4.3) the robustness of the entire scheme. We *do not* give security proofs in what follows, but point to candidate constructions for which such proofs may be easier to derive: (1) OCB, a final round CAESAR candidate introduced in [RBB03] computes the ciphertext and the tag in parallel; this makes the scheme close to the EaM composition pattern, with an additional incremental value Δ injected before calling the underlying ideal encryption procedure. (2)

Deoxys [JNPS16] is another CAESAR finalist. Deoxys-I is *nonce-respecting* (the user ensures the nonce is not reused under the same key K) and is similar to the tweakable block-cipher generalization of OCB. Deoxys-II follows the SCT mode [PS16], allows reusing a nonce under the same key and follows an EtM design. We leave a provable security treatment of the robustness amongst CAESAR candidates to future work.

To instantiate the components in Theorem 4.3, we start by observing that randomizing a CROB-secure PRF gives a pseudorandom MAC that is CROB secure. Indeed, a successful CROB adversary against this randomized PRF outputs a tuple (T, K_1, M_1, K_2, M_2) with $T = (R, Y)$ such that $\mathsf{PRF}(K_1, M_1||R) = Y = \mathsf{PRF}(K_2, M_2||R)$, which means $(Y, K_1, M_1||R, K_2, M_2||R)$ wins the CROB game against PRF.

An analogous route for directly building a CROB secure encryption scheme from a CROB secure PRF does *not* go through as the decryption algorithm of such schemes would never return $\perp$. However, by using a *common* PRF in both the encryption and MAC components, we safely reuse the keys across encryption and MAC. More precisely, given a CROB-secure PRF, the following scheme is both CROB and AE secure

$$\begin{aligned}\mathsf{Enc}(K, M; R) &:= (R, \mathsf{PRF}(K, R) \oplus M, \mathsf{PRF}(K, \mathsf{PRF}(K, R) \oplus M)))\\ \mathsf{Dec}(K, (R, C, T)) &:= \text{ if } \mathsf{PRF}(K, C) = T \text{ return } \mathsf{PRF}(K, R) \oplus C \text{ else return } \perp .\end{aligned}$$

By our theorem above, this scheme is CROB as long as the PRF is CROB. An alternative and practical route for achieving robustness uses a random oracle to instantiate the MAC as it can be easily shown to be CROB and also allows secure reuse of keys with any scheme.

The above raises the question if robustness can be achieved *without key reuse* or random oracles. Such an approach is sometimes recommended as it allows for modular proofs of AE security. Below we give a transform akin to EtM that also *authenticates the encryption key* and which results in a scheme that is both AE and CROB secure. We provide the details of the transform in Figure 4.11.

$\overline{\mathsf{KGen}}(1^\lambda)$:	$\overline{\mathsf{Enc}}((K_e, K_m), M)$:	$\overline{\mathsf{Dec}}((K_e, K_m), (C\|\|T))$:
$K_e \leftarrow\!\!\$\ \mathsf{KGen}_e(1^\lambda)$	$C \leftarrow\!\!\$\ \mathsf{Enc}(K_e, M)$	if $\mathsf{Ver}(K_m, (C\|\|K_e), T) = 0$ return $\perp$
$K_m \leftarrow\!\!\$\ \mathsf{KGen}_m(1^\lambda)$	$T \leftarrow\!\!\$\ \mathsf{Tag}(K_m, (C\|\|K_e))$	$M \leftarrow \mathsf{Dec}(K_e, C)$
return (K_e, K_m)	return (C, T)	return M

Figure 4.11: The modified EtM transform that authenticates the encryption key via a collision-resistant MAC.

Theorem 4.4. *Suppose* $\mathsf{AE} = (\mathsf{KGen}_e, \mathsf{Enc}, \mathsf{Dec})$ *is* IND$ *secure (see Figure 4.1) and* $\mathsf{MAC} = (\mathsf{KGen}_m, \mathsf{Tag}, \mathsf{Ver})$ *is pseudorandom. Then the scheme* $\overline{\mathsf{AE}} = (\overline{\mathsf{KGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ *in Figure 4.11 is* AE *secure. Furthermore, this scheme is* CROB *secure if* MAC *is collision resistant.*

Theorem 4.4. For CROB, consider an adversary that outputs $((C||T), (K_e||K_m), (K'_e||K'_m))$ such that $(C||T)$ decrypts to valid messages under both keys. Then the tag T must also verify under both K_m and K'_m. However, this constitutes an attack on the collision resistance of MAC unless $K_m = K'_m$ and $K_e = K'_e$.

For AE security, we follow the standard path as follows. Let Game_0 be the AE with real procedure. In Game_1, we compute T in the ENC procedure by replacing T with random bit strings and also replace the DEC procedure with the $\perp$ procedure. We can bound the

difference between Game_0 and Game_1 using a direct reduction to the pseudorandomness of MAC: $\Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_1^{\mathcal{A}}] \leq \mathsf{Adv}^{\$\text{uf}}_{\mathsf{MAC},\mathcal{B}_1}(\lambda)$. In Game_2 we replace the ciphertext components in the outputs of the ENC procedure with random strings. Again, using a reduction to the IND$ security of the AE scheme, we bound the difference between games Game_1 and Game_2 by: $\Pr[\text{Game}_1^{\mathcal{A}}] - \Pr[\text{Game}_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\text{ind-\$}}_{\mathsf{AE},\mathcal{B}_2}(\lambda)$. Finally, Game_2 is the AE game with fake procedures, which translates to: $\mathsf{Adv}^{\text{ae}}_{\overline{\mathsf{AE}},\mathcal{A}}(\lambda) = \Pr[\text{Game}_0^{\mathcal{A}}] - \Pr[\text{Game}_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\$\text{uf}}_{\mathsf{MAC},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{ind-\$}}_{\mathbf{SE},\mathcal{B}_2}(\lambda)$. □

4.4.1 The Symmetric ABN Transform

The starting point for our second construction is the transform introduced by Abdalla et al. [ABN10] to convert any PKE scheme into one that is also completely robust as shown in [FLPQ13]. Roughly speaking in the ABN transform one commits to the public key during encryption, encrypts the decommitment along with the plaintext, and includes the commitment as part of the ciphertext. The commitment is then checked against the public key in the decryption algorithm. The transform is shown in Figure 4.12. ABN relies on a commitment scheme $(\mathsf{CPG}, \mathsf{Com}, \mathsf{Ver})$ and operates in the CRS model via a common parameter-generation algorithm CPG.

$\overline{\mathsf{PKSetup}}(1^\lambda)$:
$\mathsf{crs} \leftarrow\!\!\$\ \mathsf{CPG}(1^\lambda)$
return crs

$\overline{\mathsf{PKKGen}}(1^\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKKGen}(1^\lambda)$
return $(\mathsf{pk}, \mathsf{sk})$

$\overline{\mathsf{PKEnc}}(\mathsf{crs}, \mathsf{pk}, M)$:
$(\mathsf{com}, \mathsf{dec}) \leftarrow\!\!\$\ \mathsf{Com}(\mathsf{crs}, \mathsf{pk})$
$C \leftarrow\!\!\$\ \mathsf{PKEnc}(\mathsf{pk}, (M, \mathsf{dec}))$
return (C, com)

$\overline{\mathsf{PKDec}}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}, (C, \mathsf{com}))$:
$(M, \mathsf{dec}) \leftarrow \mathsf{PKDec}(K, C)$
if $\mathsf{Ver}(\mathsf{crs}, \mathsf{pk}, \mathsf{com}, \mathsf{dec})$:
  return M
return $\perp$

Figure 4.12: The ABN transform [ABN10] for public-key encryption.

We ask if an analogue of ABN, perhaps in the CRS model, can also be formulated for symmetric encryption. In this setting, there is no public key and a natural alternative would be to commit to the secret key instead. This, however, results in a *key-dependent message* being encrypted as the decommitment dec is computed based on the encryption key K. Furthermore, the commitment string com must be pseudorandom to accomplish AE security.

One can attempt to adapt the ABN transform as follows. First, use a commitment scheme with pseudorandom commitments. Any *collision-resistant* PRF is equivalent to such a commitment scheme, where $\mathsf{crs} = \varepsilon$ (assuming the PRF does not use a CRS) and $\mathsf{Com}(M||K)$ outputs $(\mathsf{PRF}(K, M), K)$ as the $(\mathsf{com}, \mathsf{dec})$ pair. The verification algorithm simply checks the commitment by recomputing the PRF using K and M. This scheme is computationally hiding down to the pseudorandomness of PRF. Furthermore, it is computationally binding down to its collision resistance. This technique still does not resolve the key-dependency issue. Although in this scheme the decommitment string is merely a random PRF key independent of the encryption key, a *circular* dependency between the encryption key and the PRF key exists, which prevents a proof from going through. (Recall that in the public-key setting this issue does not arise as the public key is a key-dependent value that is available "for free.")

To fix these issues, we compute a string that acts as a "public labelling" of the encryption key, and which does not hurt the security of the scheme. We first expand K using a PRG,

use its left-half in encryption, and commit to its right-half as the public labelling. For this, we must, however, ensure that different keys give always rise to different public labellings. This can be achieved if the PRG is collision resistant (for example injective) on the right-half of outputs. Such PRGs can be based on one-way permutations via Yao's transform [Yao82]. Indeed, assuming π is a one-way permutation and HC is a hardcore predicate for it [GL89], we get a right-injective PRG via

$$\mathsf{PRG}(x) := \mathsf{HC}(x)||\mathsf{HC}(\pi(x))||\dots||\mathsf{HC}(\pi^{|x|-1}(x))||\pi^{|x|}(x) \ .$$

Observe the last part of the output of this PRG is a permutation, which provides the required injectivity; this results in the transform shown in Figure 4.13.

$\overline{\mathsf{KGen}}(1^\lambda)$:	$\overline{\mathsf{Enc}}(K_e, M)$:	$\overline{\mathsf{Dec}}(K_e, (C\|\|T))$:
$K_e \leftarrow\!\!\$\ \mathsf{KGen}_e(1^\lambda)$	$K_m \leftarrow\!\!\$\ \mathsf{KGen}_m(1^\lambda)$	$(K_e^1\|\|K_e^2) \leftarrow \mathsf{PRG}(K_e)$
return K_e	$(K_e^1\|\|K_e^2) \leftarrow \mathsf{PRG}(K_e)$	$(M\|\|K_m) \leftarrow \mathsf{Dec}(K_e^1, C)$
	$C \leftarrow\!\!\$\ \mathsf{Enc}(K_e^1, (M\|\|K_m))$	if $\mathsf{Ver}(K_m, (C\|\|K_e^2), T) = 0$
	$T \leftarrow\!\!\$\ \mathsf{Tag}(K_m, (C\|\|K_e^2))$	return $\perp$
	return $(C\|\|T)$	return M

Figure 4.13: The modified EtM transform for obtaining CROB security.

Theorem 4.5. *Let* $\mathsf{AE} = (\mathsf{KGen}_e, \mathsf{Enc}, \mathsf{Dec})$ *be* IND$ *secure,* $\mathsf{MAC} = (\mathsf{KGen}_m, \mathsf{Tag}, \mathsf{Ver})$ *be pseudorandom, and suppose* PRG *is secure. Then the scheme* $\overline{\mathsf{AE}} = (\overline{\mathsf{KGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ *in Figure 4.13 is* AE *secure. Furthermore, this scheme is* CROB *secure as long as* MAC *is collision resistant and* PRG *is right collision resistant.*

Theorem 4.5. Suppose that an adversary computes a ciphertext $(C||T)$ that decrypts correctly under two keys $K_e \neq K_e'$. The fact that $K_e \neq K_e'$ together with the right collision resistance of PRG implies that $K_e^2 \neq K_e'^2$. Then, this can be used to break the collision resistance of MAC using the pair $(K_m, (C||K_e^2))$ and $(K_m', (C||K_e'^2))$ where K_m and K_m' are computed by decrypting C using the left halves K_e^1 and $K_e'^1$ of the PRG output, respectively.

AE security can be proven in the standard way as follows. Let Game_0 be the AE game with respect to the real encryption and decryption oracles. In Game_1, we replace the outputs of the PRG with truly random bit strings. This transition can be justified using the security of PRG: $\Pr[\mathrm{Game}_0^{\mathcal{A}}] - \Pr[\mathrm{Game}_1^{\mathcal{A}}] \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{B}_1}(\lambda)$. In Game_2 we replace T with random tags and decryption with the $\perp$ oracle. A direct reduction to \$UF security of the MAC can be used to bound this transition: $\Pr[\mathrm{Game}_1^{\mathcal{A}}] - \Pr[\mathrm{Game}_2^{\mathcal{A}}] \leq \mathsf{Adv}^{\$\mathrm{uf}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda)$. In Game_3 we replace C with random strings via the IND\$ security of the AE. Now note that Game_3 corresponds to the AE game concerning the fake encryption and decryption oracles: $\Pr[\mathrm{Game}_2^{\mathcal{A}}] - \Pr[\mathrm{Game}_3^{\mathcal{A}}] \leq \mathsf{Adv}^{\mathrm{ind}\$}_{\mathsf{AE},\mathcal{B}_3}(\lambda)$. □

One advantage of the second transform is that it only relies on the pseudorandomness of MAC with freshly generated keys; this in turns allows for a simple instantiation of it. For a right collision-resistant PRG, let

$$\mathsf{PRG}(K) = \mathsf{PRG}_0(K)||\mathsf{PRG}_1(K) \text{ with } (\ell_0, \ell_1) := (|\mathsf{PRG}_0(K)|, |\mathsf{PRG}_1(K)|) \ .$$

Then we compute a MAC on a (hashed) message M with $|M| = \ell_0$ as:

$$\mathsf{Tag}(K, M) := (M||0^{\ell_1}) \oplus (\mathsf{PRG}_0(K)||\mathsf{PRG}_1(K)) \ .$$

The collision resistance of this MAC follows from the fact that the right (and collision-resistant) half of PRG is output in clear.

4.5 Robust Signature Schemes

Signature schemes are the public-key counterparts of message authentication codes. Informally, a robust signature scheme shall not allow for a verifier to validate a signature under multiple, distinct verification keys. We commence by presenting the security definition for digital signatures in Section 4.5.1 and then provide generic transforms for converting any DS scheme into a complete-robust one. Along the way, we define the intermediate notions of XROB and KROB, in the spirit of the ones proposed for message authentication codes in Section 4.2.

4.5.1 Robustness for Digital Signatures

For the case of digital signature schemes, we introduce two main security notions, which we denote strong and complete robustness. The winning condition remains the same in both experiments, of obtaining a signature/message pair in such a way that it verifies under both public keys. In the SROB experiment, two signing oracles under $\mathsf{sk}_1, \mathsf{sk}_2$ are given to the adversary, while a CROB adversary generates its intrinsic keys for accomplishing essentially the same break.

$\underline{\text{SROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)}$:	$\underline{\text{CROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)}$:
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	$(\mathsf{pk}_1, \mathsf{pk}_2, \sigma, M) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$	if $\mathsf{pk}_1 = \mathsf{pk}_2$:
$(M, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\text{Sign}_{\mathsf{sk}_1}(\cdot), \text{Sign}_{\mathsf{sk}_2}(\cdot)}(1^\lambda, \mathsf{pk}_1, \mathsf{pk}_2)$	return 0
if $\mathsf{Ver}(\mathsf{pk}_1, \sigma, M) = 1 \wedge$	if $\mathsf{Ver}(\mathsf{pk}_1, \sigma, M) = 1 \wedge$
$\mathsf{Ver}(\mathsf{pk}_2, \sigma, M) = 1$:	$\mathsf{Ver}(\mathsf{pk}_2, \sigma, M) = 1$:
return 1	return 1
return 0	return 0

Figure 4.14: Games defining strong robustness SROB (left) and complete robustness CROB (right) for a digital signature scheme DS. We assume a negligible probability of sampling $\mathsf{pk}_1 = \mathsf{pk}_2$ in the SROB game.

Definition 4.1 (SROB and CROB Security)**.** *Let* DS *be a digital signature scheme. We say* DS *achieves complete robustness if the advantage of any* PPT *adversary* $\mathcal{A}$ *against the CROB game depicted in Figure 4.14 (right side) is negligible:*

$$\mathsf{Adv}^{\text{crob}}_{\mathcal{A},\mathsf{DS}}(\lambda) := \Pr\left[\text{CROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda) = 1\right]$$

SROB*-security is defined similarly, the* $\text{SROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)$ *game being defined in Figure 4.14 (left side).*

Notice the *difference* to the classical unforgeability game where the adversary obtains signatures issued under the *same* secret key. We prove any EUF-scheme is implicitly strong-robust, and show there exist signature schemes that fail to achieve complete robustness (thus providing a separation between the two).

Remark 4.1 (Comparison with Unambiguity)**.** *Bellare and Duan [BD09] had described, earlier but in a different context, a notion of digital signature* unambiguity*.*

As stated in [BD09], "Unambiguity can be viewed as a signature analogue of the robustness property of anonymous encryption defined in [ABN10]. [...] Unambiguity [...] can be viewed as preventing forgery under an adversarially-modified verification key, something not part of the normal definition of a signature." The original motivation for unambiguity stems from the design of *partial signatures*.

It is natural to wonder whether unambiguity (UNAMB) coincides with either notion of signature robustness discussed above. Since unforgeability does not imply unambiguity, and since any partial signature scheme is a signature scheme, we have SROB $\neq$ UNAMB. However, it turns out that the definition UNAMB (for partial signatures) is naturally extended to signatures and matches CROB.

4.5.2 Intermediate Notions

To keep a symmetry with the existing works in public and symmetric key settings [FLPQ13; FOR17], intermediate notions of robustness such as *key-less* (KROB) or *mixed* (XROB) robustness can be introduced. To give a flavour, in a KROB game, apart from M, an adversary issues $(\mathsf{sk}_1, M_1, R_1)$ on one side, and $(\mathsf{sk}_2, M_2, R_2)$ on the other side, R_i standing for the random coins used to sign M_i under sk_i. One can then show trivial relations between these notions, with details being provided below. For brevity, in this section we only use the SROB and CROB notions. On a different note, we point out that more interesting *intermediate* notions result if partial access to a key (i.e., revealing random positions in the binary representation of the keys) is provided to an adversary [HS09].

Definition 4.2 (XROB and KROB Security)**.** *Let* DS *be a digital signature scheme. Complete robustness is defined as the advantage of any* PPT *adversary* $\mathcal{A}$ *against the XROB game depicted in Figure 4.15 (right side):*

$$\mathsf{Adv}^{\mathrm{xrob}}_{\mathcal{A},\mathsf{DS}}(\lambda) \coloneqq \Pr\left[\mathrm{XROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda) = 1\right] .$$

KROB*-security is defined similarly, the* $\mathrm{KROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)$ *game being defined in Figure 4.15 (right side).*

Lemma 4.1. *Any* CROB*-secure digital signature scheme is also* XROB*-secure. Any* XROB*-secure digital signature scheme is also* KROB*-secure.*

Lemma 4.1. For the first case, we take the contrapositive. Assuming an XROB adversary that returns $(\sigma_1, M, \mathsf{pk}_1, \mathsf{sk}_2, R_2)$, the reduction builds a public key pk_2 via $\mathrm{PKGen}(\mathsf{sk}_2)$. The winning CROB tuple is, therefore $(\sigma_1, M, \mathsf{pk}_1, \mathsf{pk}_2)$. For the second part, assuming a KROB adversary returning $(M, \mathsf{sk}_1, R_1, \mathsf{sk}_2, R_2)$, the reduction generates $\mathsf{pk}_1 \leftarrow\!\!\$\ \mathrm{PKGen}(\mathsf{sk}_1)$ and returns the XROB winning tuple $(\sigma, M, \mathsf{pk}_1, \mathsf{sk}_2, R_2)$, where $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_1, M; R_1)$. □

4.5.3 Implications and Separations

Proposition 4.4. *Let* DS *be a* CROB*-secure digital signature scheme. Then* DS *is also* SROB*-secure, the advantage of breaking the strong robustness game being bounded as follows:*

$$\mathsf{Adv}^{\mathrm{srob}}_{\mathcal{A},\mathsf{DS}}(\lambda) \leq \mathsf{Adv}^{\mathrm{crob}}_{\mathcal{R},\mathsf{DS}}(\lambda) .$$

$\underline{\text{XROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)}$:	$\underline{\text{KROB}^{\mathcal{A}}_{\mathsf{DS}}(\lambda)}$:
1. $(\sigma_1, M, \mathsf{pk}_1, \mathsf{sk}_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$	1. $(M, \mathsf{sk}_1, R_1, \mathsf{sk}_2, R_2) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
2. if $\mathsf{pk}_1 = \text{PKGen}(\mathsf{sk}_2)$:	2. if $\text{PKGen}(\mathsf{sk}_1) = \text{PKGen}(\mathsf{sk}_2)$:
3. return 0	3. return 0
4. $\sigma_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, M; R_2)$	4. $\sigma_1 \leftarrow \mathsf{Sign}(\mathsf{sk}_1, M; R_1)$
5. $b \leftarrow \mathsf{Ver}(\mathsf{pk}_1, \sigma, M)$	5. $\sigma_2 \leftarrow \mathsf{Sign}(\mathsf{sk}_2, M; R_2)$
6. if $b = 1 \ \wedge \sigma_1 = \sigma_2$:	6. if $\sigma_1 = \sigma_2$:
7. return 1	7. return 1
8. return 0	8. return 0

Figure 4.15: Games defining mixed-robustness XROB (left) and keyless-robustness KROB (right) for a digital signature scheme DS. We assume that We assume that given a secret-key sk, there exists a procedure PKGen for generating a public-key.

Proposition 4.4. Suppose DS is not SROB-secure. Let $\mathcal{A}$ be a PPT adversary that wins the SROB game with an advantage at most ϵ_{SROB}. We construct a PPT adversary $\mathcal{R}$ against the CROB game as follows: (1) sample two pairs of keys $(\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2)$ using $\mathsf{KGen}(1^\lambda)$; (2) $\mathcal{R}$ publishes $\mathsf{pk}_1, \mathsf{pk}_2$ and constructs the signing oracles $\mathsf{Sign}_{\mathsf{sk}_1}(\cdot)$ and $\mathsf{Sign}_{\mathsf{sk}_2}(\cdot)$; (3) $\mathcal{R}$ runs $\mathcal{A}$ w.r.t. signing oracles and public-keys to obtain (M, σ); (4) $\mathcal{R}$ constructs the tuple (pk_1, pk_2, σ, M) and outputs it. We obtain that $\mathsf{Adv}^{\text{srob}}_{\mathcal{R},\mathsf{DS}}(\lambda) \leq \mathsf{Adv}^{\text{crob}}_{\mathcal{A},\mathsf{DS}}(\lambda)$. □

Of interest, is a minimal level of robustness achieved by any digital signature scheme. It turns out that SROB is accomplished and we formalize this in the following lemma.

Lemma 4.2. *Any* EUF*-secure digital signature scheme* DS *is* SROB*-secure. The advantage of breaking the* SROB *game is bounded by the advantage of breaking the* EUF *game:* $\mathsf{Adv}^{\text{srob}}_{\mathcal{A},\mathsf{DS}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\text{euf}}_{\mathcal{R},\mathsf{DS}}(\lambda)$.

Lemma 4.2. Let $\mathcal{A}$ be a PPT adversary against the strong robustness game. Let $\mathcal{R}$ stand for an adversary against the unforgeability of the digital signature. We assume without loss of generality that $\mathcal{A}$: (1) never queries a "winning" message M to the second signing oracle after it has been signed by the first oracle (since it can check it right away) and (2) it never queries a "winning" message M to the first oracle after it has been signed by the second oracle (for the same reason). We present the reduction in Figure 4.16 and describe it below:

1. The EUF game proceeds by sampling $(\mathsf{sk}_1, \mathsf{pk}_1)$ and builds a signing oracle $\text{Sign}_{\mathsf{sk}_1}(\cdot)$.
2. The reduction $\mathcal{R}$ is given pk_1 and oracle access to the $\text{Sign}_{\mathsf{sk}_1}(\cdot)$. $\mathcal{R}$ samples uniformly at random $(\mathsf{sk}_2, \mathsf{pk}_2)$ via DS.KGen and constructs a second signing oracle $\text{Sign}_{\mathsf{sk}_2}(\cdot)$.
3. $\mathcal{R}$ runs $\mathcal{A}$ w.r.t. the two $(\mathsf{pk}_1, \mathsf{pk}_2)$ and the corresponding signing oracles $\text{Sign}_{\mathsf{sk}_1}(\cdot)$, $\text{Sign}_{\mathsf{sk}_2}(\cdot)$. $\mathcal{R}$ keeps track of the queried messages to each oracle.
4. $\mathcal{A}$ returns a pair (σ, M) which verifies under both public keys with probability ϵ_{SROB}, s.t. M has been queried to either $\text{Sign}_{\mathsf{sk}_1}$ or $\text{Sign}_{\mathsf{sk}_2}$ but not to both.
5. $\mathcal{R}$ returns (σ, M). If $M \in \mathsf{Sign}_{\mathsf{sk}_1}(\cdot).\text{SignedMessages}()$, $\mathcal{R}$ aborts and reruns $\mathcal{A}$. With probability $\frac{1}{2}$, M was not queried before to $\mathsf{Sign}_{\mathsf{sk}_1}(\cdot)$. The tuple (σ, M) wins the EUF game w.r.t. $(\mathsf{pk}_1, \mathsf{sk}_1)$ with probability $\geq \frac{1}{2} \cdot \epsilon_{\text{SROB}}$.

Algorithm $\mathcal{R}_{\mathcal{A}}(\lambda, \mathsf{pk}_1, \mathrm{Sign}_{\mathsf{sk}_1}(\cdot))$:
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$
build $\mathrm{Sign}_{\mathsf{sk}_2}(\cdot)$
$(M, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{Sign}_{\mathsf{sk}_1}(\cdot), \mathrm{Sign}_{\mathsf{sk}_2}(\cdot)}(\mathsf{pk}_1, \mathsf{pk}_2)$
if $M \in \mathsf{Sign}_{\mathsf{sk}_1}(\cdot).\mathrm{SignedMessages}()$
    abort
return (M, σ)

Figure 4.16: The reduction $\mathcal{R}$ in Lemma 4.2.

Thus, the reduction (Figure 4.16) shows that the advantage of winning SROB is bounded by the advantage of breaking EUF, which completes the proof. □

Verifiable random functions [MRV99] are PRFs where proofs for the correctness of their computations are issued. They generically imply (deterministic) digital signatures; thus we obtain the subsequent corollary.

Corollary 4.1. *Any verifiable random function is* SROB*-secure. The advantage of breaking the* SROB *game is bounded by the advantage of breaking the pseudorandomness game of the VRF as follows*

$$\mathsf{Adv}^{\mathrm{srob}}_{\mathcal{R},\mathsf{DS}}(\lambda) \leq \mathsf{Adv}^{\mathrm{RAND}}_{\mathcal{A},\mathsf{VRF}}(\lambda) \ .$$

Proof. Trivially follows from the fact that any VRF is a (deterministic) signature scheme, which is SROB-secure. □

We also show a separation between the SROB and CROB, by pointing to a signature scheme that is not CROB secure (but already SROB).

Proposition 4.5. *There exist DS schemes that are not* CROB*-secure.*

Proposition 4.5. We provide a simple counterexample as follows. Consider the digital signature scheme in [BB08]:

- KGen: selects uniformly at random $g_1 \leftarrow\!\!\$\ \mathbb{G}_1, g_2 \leftarrow\!\!\$\ \mathbb{G}_2$ and $(x, y) \leftarrow\!\!\$\ \mathbb{Z}_p^2$. Set $\mathsf{sk} \leftarrow (x, y)$ and $\mathsf{pk} \leftarrow (g_1, g_2, g_2^x, g_2^y, e(g_1, g_2))$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing[3].
- Sign: given a message M, sample $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute $\sigma \leftarrow g_1^{1/(x+M+yr)}$. Note that with overwhelming probability, $x + M + yr \neq 0 \bmod p$, where p is the order of $\mathbb{G}_1$. The signature is the pair (σ, r).
- Verify: check that $e\left(\sigma, g_2^x \cdot g_2^M \cdot (g_2^y)^r\right) \stackrel{?}{=} e(g_1, g_2)$.

To win the CROB game, an adversary $\mathcal{A}$ proceeds as follows:

1. $\mathcal{A}$ samples a key-pair: $\mathsf{sk} \leftarrow\!\!\$\ (x, y); \mathsf{pk} \leftarrow (g_1, g_2, g_2^x, g_2^y, e(g_1, g_2))$ and a message $M \in \mathbb{Z}_p$.
2. $\mathcal{A}$ samples $r \leftarrow\!\!\$\ \mathbb{Z}_p$ and computes σ under sk_1. Since g'_1 can be written as g_1^t, $\mathcal{A}$ sets t, x', y' such that $1/(x + M + yr) = t/(x' + M + y'r)$ (equate the exponents to obtain the same σ corresponding to M). This can be done by assigning random values to x', y' and setting $t \leftarrow (x' + M + y'r)/(x + M + yr)$.

[3] See for instance [BB08] for the definition and usage of a cryptographic pairing.

3. $\mathcal{A}$ sets $\mathsf{sk}' \leftarrow (x', y')$; $\mathsf{pk}' \leftarrow ({g'}_1, {g'}_2, {g'}_2^{x'}, {g'}_2^{y'}, e({g'}_1, {g'}_2))$, for some uniformly sampled generator ${g'}_2 \leftarrow\!\!\$\ \mathbb{G}_2$.

4. Finally, observe that (σ, r) verifies under $(\mathsf{sk}_1, \mathsf{pk}_1)$ through the correctness of the signature scheme, but also under $(\mathsf{pk}_2, \mathsf{sk}_2)$, since

$$e\left(g_1^{t/(x'+M+y'r)}, {g'}_2^{x'} \cdot {g'}_2^{M} \cdot ({g'}_2^{y'})^r\right) = e(g_1^t, {g'}_2) \ .$$

   $\mathcal{A}$ halts and returns $(\mathsf{pk}, \mathsf{pk}', (\sigma, r), M)$. Note that $\mathcal{A}$ runs in probabilistic polynomial time.

□

4.5.4 Generic Transform

4.5.4.1 Robust Digital Signatures

We show a generic transform similar in spirit to the original work of Abdalla, Bellare, and Neven [ABN10; ABN18] in the context of digital signatures. For a digital signature scheme, we benefit from the fact that pk acts as an "immutable" value to which one can easily commit to while signing a message. Thus, checking if a message verifies under another public key implicitly breaks the binding property of the commitment scheme. For simplicity, we use a hash instead of a commitment scheme.

$\underline{\overline{\mathsf{KGen}}(1^\lambda)}$:
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{DS.KGen}(1^\lambda)$
$\overline{\mathsf{pk}} \leftarrow \mathsf{pk}$
$\overline{\mathsf{sk}} \leftarrow \mathsf{sk}$
return $(\overline{\mathsf{sk}}, \overline{\mathsf{pk}})$

$\underline{\overline{\mathsf{Sign}}(\overline{\mathsf{sk}}, M)}$:
$\mathsf{sk} \leftarrow \overline{\mathsf{sk}}$
$\sigma_1 \leftarrow\!\!\$\ \mathsf{DS.Sign}(\mathsf{sk}, M)$
$\sigma_2 \leftarrow \mathsf{H}(\mathsf{pk})$
$\overline{\sigma} \leftarrow (\sigma_1, \sigma_2)$
return $\overline{\sigma}$

$\underline{\overline{\mathsf{Ver}}(\overline{\mathsf{pk}}, \overline{\sigma}, M)}$:
$\mathsf{pk} \leftarrow \overline{\mathsf{pk}}$
$(\sigma_1, \sigma_2) \leftarrow \overline{\sigma}$
return $\mathsf{DS.Ver}(\mathsf{pk}, \sigma_1) = 1 \wedge \sigma_2 \stackrel{?}{=} \mathsf{H}(\mathsf{pk})$

$\underline{\overline{\mathsf{Setup}}(1^\lambda)}$:
$K \leftarrow \mathsf{H.KGen}(1^\lambda)$; $\mathsf{H} \leftarrow \mathsf{H}_K$; return H

Figure 4.17: A generic transform that turns any digital signature scheme DS into one that is, in addition, CROB-secure. The (publicly available) collision-resistant hash function H can be based on claw-free permutations in the standard model, as shown in the seminal work of Damgård [Dam88]. It is used as a commitment to the public-key.

Lemma 4.3. *Let* DS *be an* EUF*-secure digital signature scheme. Let* H *denote a collision-resistant hash function. The digital signature* $\overline{\mathsf{DS}}$ *obtained through the transform depicted in Figure 4.17 is* CROB*-secure.*

Lemma 4.3. We prove both the unforgeability and the complete robustness of the newly obtained construction:

Unforgeability. Assume the existence of a PPT adversary $\mathcal{A}$ against $\overline{\mathsf{DS}}$. We build an adversary $\mathcal{R}$ against the EUF of the underlying DS. The unforgeability experiment EUF for DS samples $(\mathsf{pk}, \mathsf{sk})$ and constructs a signing oracle under sk, which is given to $\mathcal{R}$. $\mathcal{R}$ is given a collision-resistant hash function H and builds its own signing oracle $\overline{\mathsf{Sign}}$; when queried,

$\overline{\mathsf{Sign}}$ returns the output of Sign concatenated to the value of $\mathsf{H}(\mathsf{pk})$. When $\mathcal{A}$ replies with $(\overline{\sigma}, M)$, it must be the case that $\mathsf{Ver}(\mathsf{pk}, \sigma, M)$ passes, which breaks EUF for DS. Thus we conclude that: $\mathsf{Adv}^{\mathrm{euf}}_{\mathcal{A},\overline{\mathsf{DS}}}(\lambda) \leq \mathsf{Adv}^{\mathrm{euf}}_{\mathcal{R},\mathsf{DS}}(\lambda)$.

CROB. To show robustness, we rely on the collision-resistance of H. The CROB game in Figure 4.14 specifies that the adversary $\mathcal{A}$ against the CROB game finds $\mathsf{pk}_1 \neq \mathsf{pk}_2$ such that $\overline{\mathsf{Ver}}$ passes. The latter implies $\mathsf{H}(\mathsf{pk}_1) = \mathsf{H}(\mathsf{pk}_2)$, trivially breaking the collision-resistance of H, giving us: $\mathsf{Adv}^{\mathrm{crob}}_{\mathcal{A},\overline{\mathsf{DS}}}(\lambda) \leq \mathsf{Adv}^{\mathrm{cr}}_{\mathcal{R},\mathsf{H}}(\lambda)$.

□

Chapter 5

WBC - Definitions and Relations

In this chapter, we investigate the feasibility of white-box compilers for pseudorandom permutations obtained on top of (multi-input) functional encryption. We begin by presenting a set of simple theoretical implications, which suggest that a central notion, dubbed one-wayness, is instrumental in achieving UBK-secure compilers. We also look into specific constructions of FE schemes, pointing out that some of them suffice in obtaining UBK-secure implementations.

Chapter Organization. A brief motivation behind the need for white-box implementations has been discussed in Section 1.1.1. For ease of exposition, we provide a short roadmap of this chapter. The main results of this part are pictorially described in Figure 5.1.

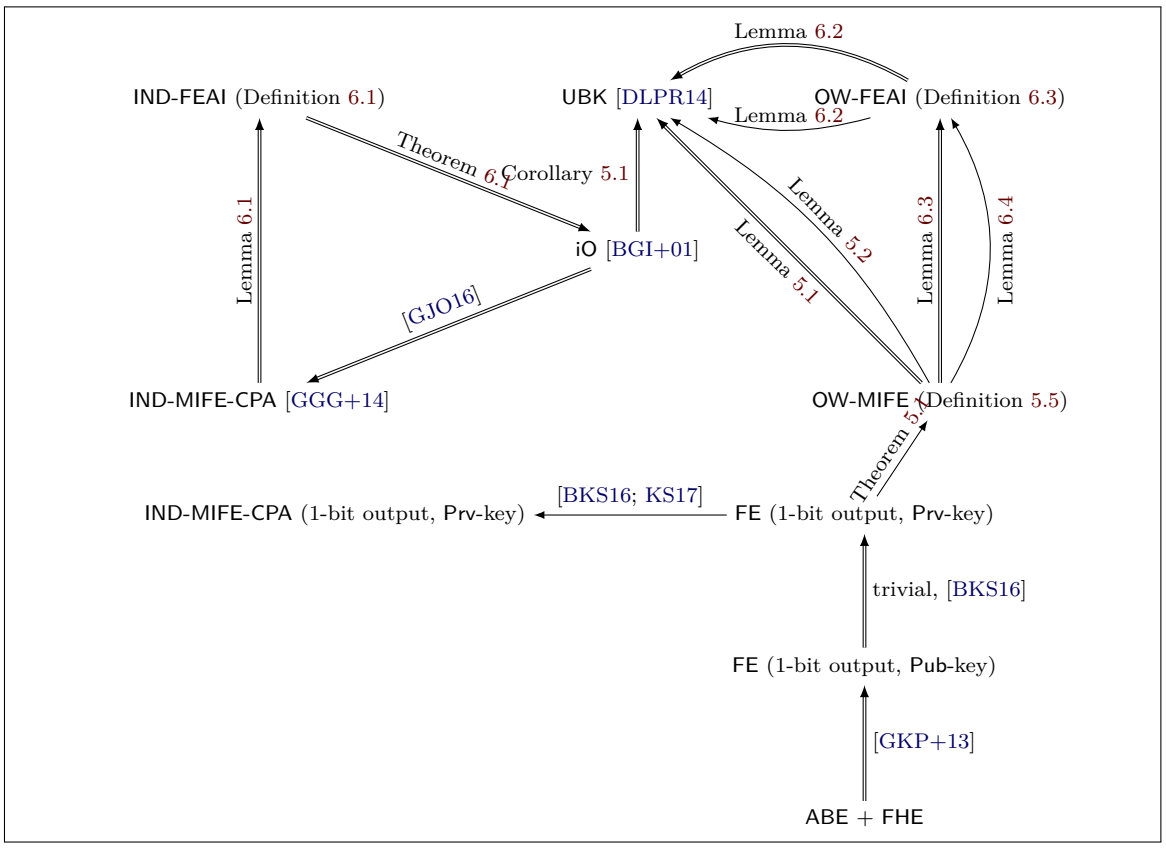

Figure 5.1: A graphical depiction of the main results in this chapter. Double-arrows originating in an "FE" node denote that "FE" is considered in the public key setting. Single arrows originating in an "FE" node denote that the FE scheme is in the private-key setting.

In Section 5.1, we provide the standard definitions related to white-box cryptography as well as multi-input functional encryption (MIFE), a notion that proves instrumental to the incoming steps[1]. We extend the classical notion of *one-wayness* in Section 5.2, by introducing simple but powerful *one-wayness* definitions for multi-input functional encryption schemes (OW-MIFE) in both the public and private-key settings (Section 6.1). We show both settings are sufficient to achieve UBK-secure implementations for some pseudorandom permutation f. Proving that OW-MIFE is sufficient to construct UBK implementations for one-way permutations is relatively easy. Our compiler relying on public-key MIFE (two inputs suffice) works as follows: it computes $C_K \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{mpk}_1, K)$ and the functional encryption key sk_f for the publicly available f. A UBK-secure implementation of f then

[1] In the following chapter (Section 6.1), we will introduce the related concept of functional encryption with auxiliary inputs (FEAI), as well as its *indistinguishability* and *one-wayness* security notions. We show IND-FEAI schemes are sufficient to obtain indistinguishability obfuscation. We regard this as an alternative potential path for obtaining iO, a problem of independent interest.

consists of the decryption algorithm plus the variables mpk_2, sk_f and C_K. For $f(K, M)$ to be simulated, the decryption algorithm computes

$$\mathsf{MIFE.Dec}(\mathsf{sk}_f, C_K, \mathsf{MIFE.Enc}(\mathsf{mpk}_2, M))$$

which, by the correctness of the MIFE, is equivalent to $f(K, M)$.

In the private key setting, the crux idea for achieving a UBK implementation w.r.t a pseudorandom permutation is to allocate a separate slot of the MIFE scheme per *each* input bit of the plaintext M (Figure 5.2). Then, one would release MIFE encryptions of 0s and 1s for the corresponding positions. The key K used by f is encrypted separately. Then computing $f(K, M)$ is done by simply selecting the appropriate encryptions of 0s and 1s based on the binary decomposition of M and decrypt via MIFE.Dec. Moreover, we note that in a very similar way, one can obtain a simple FEAI construction (defined in Chapter 6). As regards one-wayness, we need the MIFE construction to be one-way only w.r.t the slot used to encrypt K (otherwise it is trivial to win the game).

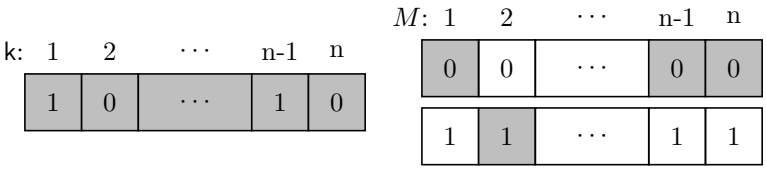

Figure 5.2: A ciphertexts is generated for the key K, as well as for each of the bits in the binary decomposition of the message.

We give a positive answer to the question of obtaining white-box implementations that are secure against adversaries mounting key-extraction attacks (UBK-security). Although relying on strong assumptions, to the best of our knowledge, this is the first time when the problem has been rigorously studied and answered. We prove there exist constructions of MIFE schemes that are one-way secure (Section 5.4). Proving the existence of such OW-MIFE schemes Theorem 5.2 in the public-key setting is done by referring to the peculiarities of the construction proposed by Goyal, Jain and O'Neill in [GJO16]. We note their scheme assumes the existence of iO and of OWF, but it is simple enough to enable a proof via a standard hybrid argument.

Along the way, we look into the power conferred by multi-input functional encryption in the private-key setting. Specifically, we investigate the generic transform introduced by Brakerski, Komargodski and Segev in [BKS16] (from now on the "BKS" transform); this builds a t-input MIFE scheme on top of a single input FE scheme for general circuits and is also function-hiding. We prove the BKS transform enjoys one-wayness assuming the original single-input scheme is one-way. Furthermore, we argue that such single input schemes can be instantiated from standard assumptions, by referring to the construction of Goldwasser et al. [GKP+13]. Care is needed in order to ensure the final transform is one-way: our proof walks through one-wayness of the required primitives, starting with the single-input FE instantiation in [GKP+13], and continuing with the function hiding transform of [BS15] and finally for the BKS transform itself.

5.1 Definitions for White-Box Cryptography

5.1.1 White-Box Implementations for Block Ciphers

This chapter focuses on white-box implementations for block-ciphers (viewed as pseudorandom permutations) due to their practical impact. Throughout the chapter we assume a key-space of the form $\mathcal{K} = \{0,1\}^{|K|}$ and a message and a ciphertext space of the form: $\mathcal{M} = \mathcal{C} = \{0,1\}^{|M|}$. A white-box compiler for a block-cipher $\mathcal{E}$, denoted by $\mathsf{C}_{\mathcal{E}}$, is a program that takes as input a key k from the key space $\mathcal{K}$ and a nonce $r \leftarrow\!\!\$\ \mathcal{R}$ for some space $\mathcal{R}$, then outputs a program embedding k – denoted $\mathsf{C.Eval}_k^r$ – where r is an option for generating diversified implementations for the same k. $\mathsf{C}_{\mathcal{E}}$ is correct if $\mathsf{C.Eval}_k^r$ exactly implements $\mathcal{E}(k,\cdot)$, namely, for any $M \in \mathcal{M}$ we have $\mathsf{C.Eval}_k^r(M) = \mathcal{E}.\mathsf{Enc}(k, M)$.

Definition 5.1 (Compilers for Block Ciphers)**.** *A compiler $\mathsf{C}_{\mathcal{E}}$ for a block cipher described in some language as $\mathcal{E}$, consists of a tuple of programs* $(\mathsf{C.Setup}, \mathsf{C.Eval})$ *– implemented algorithms – described as follows:*

- $\mathsf{C.Setup}(\mathcal{E}, K)$*: the compiler takes the description $\mathcal{E}$, together with a secret key K and a nonce r, and outputs two programs* $(\mathsf{C.Eval}_k^r, \mathsf{C}^{-1}.\mathsf{Eval}_k^r)$.
- $C \leftarrow \mathsf{C.Eval}_k^r(M)$*: the program $\mathsf{C.Eval}_k^r$ takes as input a message M and outputs a ciphertext corresponding to $\mathcal{E}.\mathsf{Enc}(K, M)$. $\mathsf{C}^{-1}.\mathsf{Eval}_k^r$ is defined similarly for decryption.*

$\mathsf{UBK}_{\mathsf{C}_{\mathcal{E}}}^{\mathcal{A}}(\lambda)$:
$R \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$
$\mathsf{C.Eval}_k^r \leftarrow \mathsf{C.Setup}(1^\lambda, K, R, \mathcal{E})$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_{\mathcal{E}}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

Figure 5.3: The unbreakability security experiment as defined in [DLPR14].

Delerablée *et al.* , [DLPR14] introduced several security notions related to white-box cryptography for symmetric encryption. As we focus on the existence of white-box cryptography, we present below the essential *unbreakability* notion.

Definition 5.2 (Unbreakability [DLPR14])**.** *Let $\mathcal{E}$ be a symmetric-key encryption scheme and $\mathsf{C}_{\mathcal{E}}$ a compiler for $\mathcal{E}$. We say that $\mathsf{C}_{\mathcal{E}}$ is unbreakable if the following advantage is negligible against any adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_{\mathcal{E}}}(\lambda) := \Pr[\mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_{\mathcal{E}}}(\lambda) = 1] \leq \textsc{Negl}\ .$$

where the security experiment UBK *is defined in Section 5.1.1.*

We note that such a definition can be naturally extended to other primitives, such as pseudorandom functions.

5.1.2 Multi-Input Functional Encryption

Functional encryption, as defined in Chapter 2, is one of the most general and abstract encryption paradigms and can be defined in multiple scenarios: for instance, the encryption key can be either public or private, or the functional key can be function hiding or function revealing. Below, we recall the definition of FE in the multi-input setting by Goldwasser *et al.* [GGG+14], in both private and public encryption settings.

5.1.2.1 MIFE - Public-Key Setting

Definition 5.3 (Public-Key MIFE) [GGG+14]**.** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in N}$ be an ensemble, where $\mathcal{F}_\lambda$ is a finite collection of n-ary functions $f : \mathcal{X}_{\lambda,1} \times \ldots \times \mathcal{X}_{\lambda,n} \to Y_\lambda$. A public-key multi-input functional encryption scheme* MIFE *for $\mathcal{F}_\lambda$ consists of four algorithms* (Setup, KGen, Enc, Dec)*:*

- *The setup algorithm* $\mathsf{Setup}(1^\lambda)$ *takes the security parameter λ in unary and outputs a master secret key* msk *and n encryption keys* $\{\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n\}$*;*
- *The encryption algorithm* $\mathsf{Enc}(\mathsf{mpk}_i, M_i)$ *takes as input an encryption key* mpk_i *and an input message $M_i \in \mathcal{X}_{\lambda,i}$, and outputs a ciphertext C_i, for some position $i \in [n]$;*
- *The functional-key derivation algorithm* $\mathsf{KGen}(\mathsf{msk}, f)$ *takes as input the description of an n-ary function $f \in \mathcal{F}_\lambda$ and outputs the corresponding functional key* sk_f*;*
- *The decryption algorithm* $\mathsf{Dec}(\mathsf{sk}_f, C_1, \ldots, C_n)$ *is a deterministic algorithm that takes as input a functional key* sk_f *and an ordered list of n ciphertext $(C_1, \ldots, C_n)$ and outputs a string y corresponding to $f(M_1, \ldots, M_n)$ or a special error symbol $\perp$.*

Any public-key MIFE *scheme is required to satisfy* ***correctness****:*

$$\Pr\left[\begin{array}{l|l} \mathsf{Dec}(\mathsf{sk}_f, & \\ \quad \mathsf{Enc}(\mathsf{mpk}_1, M_1), \ldots, & (\mathsf{msk}, \mathsf{mpk}_1, \ldots, \mathsf{mpk}_n) \leftarrow\!\$\ \mathsf{Setup}(1^\lambda) \wedge \\ \quad \mathsf{Enc}(\mathsf{mpk}_n, M_n)) = & \mathsf{sk}_f \leftarrow\!\$\ \mathsf{KGen}(\mathsf{msk}, f) \\ f(M_1, \ldots, M_n) & \end{array}\right] \in 1 - \textsc{Negl}(\lambda).$$

Indistinguishability *for multi-input functional encryption states that for any possible combinations of ("challenge ciphertexts", "adversarially computable ciphertext"), the* KGen*-queried functions return the same values: $f(ChalPlaintxt_1, AdversPlaintext) = f(ChalPlaintxt_2, AdversPlaintext)$. We define the notion for 2-input* FE *schemes and point the reader to [GGG+14, p. 9] for the full definition.*

Formally, for any set of message pairs defined over $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$ and for any PPT *adversary $\mathcal{A}$, its advantage against the game* $\mathsf{IND\text{-}MIFE\text{-}CPA}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda)$ *defined in Figure 5.4 is negligible:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}mife\text{-}cpa}}_{\mathcal{A},\mathsf{MIFE}}(\lambda) := \left|\Pr[\mathsf{IND\text{-}MIFE\text{-}CPA}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda) = 1] - \frac{1}{2}\right| .$$

5.1.2.2 MIFE - Private-Key Setting.

The private-key counterpart follows naturally, the key difference to the public-key setting being that encryption is done under msk, as the Setup generates no mpk. A subsequent change occurs in the description of the indistinguishability game that requires an adversary to interact with *encryption oracles*.

$\underline{\mathsf{IND\text{-}MIFE\text{-}CPA}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathcal{I} \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
$(\mathsf{msk}, \{\mathsf{mpk}_1, \mathsf{mpk}_2\}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]} \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}_{\mathsf{msk}}(\cdot)}(\{\mathsf{mpk}_i\}_{i\in I})$
for $i \leftarrow 1, q$:
  $C^*_{1,i} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_1, M^b_{1,i})$
  $C^*_{2,i} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}_2, M^b_{2,i})$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}^{(\cdot)}_{\mathsf{msk}}}\left(\left\{C^*_{j,i}\right\}_{j\in\{1,2\}, i\in[q]}, \{\mathsf{mpk}_i\}_{i\in I}\right)$
return $b = b' \wedge \mathrm{Valid}(\mathrm{L}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})$

$\underline{\mathrm{Valid}(\mathrm{L}, \{\mathsf{mpk}_i\}_{i\in I}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})}$:
if $\exists f \in \mathrm{L} \wedge \exists i \in [q] \wedge \exists X_1 \in \mathcal{X}_1$ s.t.:
  if $f(X_1, M^0_{2,i}) \neq f(X_1, M^1_{2,i})$:
    return 0
if $\exists f \in \mathrm{L} \wedge \exists i \in [q] \wedge \exists X_2 \in \mathcal{X}_2$ s.t.:
  if $f(M^0_{1,i}, X_2) \neq f(M^1_{1,i}, X_2)$:
    return 0
return 1

$\underline{\mathsf{KGen}_{\mathsf{msk}}(f)}$:
$\mathrm{L} \leftarrow \mathrm{L} \cup \{f\}$
return $\mathsf{KGen}(\mathsf{msk}, f)$

Figure 5.4: The (adaptive) IND-MIFE-CPA security game, as defined in [GGG+14]. In our description, we use $\overrightarrow{M}$ to denote a 2-ary message. L stands for a set of queried functions. The Valid algorithm enforces that the queried functions $f \in \mathrm{L}$ produce identical outputs when queried on (challenge message, $\cdot$) or ($\cdot$,challenge message).

Definition 5.4 (Private-Key MIFE)**.** *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in N}$ be an ensemble, where $\mathcal{F}_\lambda$ is a finite collections of n-ary functions $f : \mathcal{X}_{\lambda,1} \times \ldots \times \mathcal{X}_{\lambda,n} \to Y_\lambda$.A multi-input functional encryption scheme* MIFE *for $\mathcal{F}$ is a tuple of* PPT *algorithms* $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

- $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$*: the key generation procedure outputs a master secret key* msk*.*
- $C_i \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, i, M_i)$*: the encryption procedure takes as input a plaintext M_i and a position i – corresponding to the i-th input of a supported function f – and encrypts M_i under* msk *for position i.*
- $\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f)$*: the key-derivation procedure takes as input the description of a function f and outputs the corresponding sk_f.*
- $f(M_1, \ldots, M_n) \leftarrow \mathsf{Dec}(\mathsf{sk}_f, C_1, \ldots, C_n)$*: while attempting to decrypt under sk_f, we require an ordered list of ciphertexts as arguments $[C_1, \ldots, C_n]$ (corresponding to the positions for which they were produced); the result is the $f(M_1, \ldots, M_n)$ or a special error symbol $\perp$.*

Any private-key MIFE *scheme is required to satisfy* ***correctness****: for any messages $M_1 \in \mathcal{X}_{\lambda,1}, \ldots, M_n \in \mathcal{X}_{\lambda,n}$ and $f \in \mathcal{F}$ we have that*

$$\Pr\left[\begin{array}{l|l} \mathsf{MIFE.Dec}(\mathsf{sk}_f, & \mathsf{msk} \leftarrow\!\!\$\ \mathsf{FE.KGen}(1^\lambda) \wedge \\ \quad \mathsf{MIFE.Enc}(\mathsf{msk}, (M_1, 1)), \ldots & \mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{FE.KGen}(\mathsf{msk}, f) \\ \quad \mathsf{MIFE.Enc}(\mathsf{msk}, (M_n, n))) = & \\ f(M_1, \ldots, M_n) & \end{array}\right] \in 1 - \mathrm{NEGL}(\lambda) \ .$$

Indistinguishability*: suppose q is the number of challenge message sampled by the adversary, the* IND-MIFE-CPA *(Figure 5.5) requires the negligibility of the following advantages:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}mife\text{-}cpa}}_{\mathcal{A},\mathsf{MIFE}}(\lambda) := \left|\Pr[\mathsf{IND\text{-}MIFE\text{-}CPA}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda) = 1] - \frac{1}{2}\right| \ .$$

$\underline{\mathsf{IND\text{-}MIFE\text{-}CPA}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]} \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{KGEN}_{\mathsf{msk}}(\cdot),\mathrm{ENC}_{\mathsf{msk}}(\cdot)}(1^\lambda)$
for $i \leftarrow 1, q$:
  for $j \leftarrow 1, n$:
    $C^*_{j,i} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, j, M^b_{j,i})$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}^{(\cdot)}_{\mathsf{msk}},\mathrm{ENC}_{\{\mathsf{msk}\}}(\cdot)}\left(1^\lambda, \left\{C^*_{i,j}\right\}_{i\in[q],j\in[n]}\right)$
return $b = b' \wedge \mathrm{Valid}(\mathrm{L}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})$

$\underline{\mathrm{KGEN}_{\mathsf{msk}}(f)}$:
$\mathrm{L} \leftarrow \mathrm{L} \cup \{f\}$
return $\mathsf{KGen}(\mathsf{msk}, f)$

$\underline{\mathrm{Valid}(\mathrm{L}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})}$:
if $\exists f \in \mathrm{L}, \exists i \in [q] : f(\overrightarrow{M}_i^0) \neq f(\overrightarrow{M}_i^1)$:
  return 0
return 1

Figure 5.5: The (adaptive) IND-MIFE-CPA security experiment for multi-input functional encryption schemes in the private setting, as derived from [GGG+14]. In our description, we use $\overrightarrow{M}$ to denote an n-ary message. L stands for the list of queried functions. The Valid procedure enforces that for any combination of queried messages, the two functions return the same outputs.

***Function-Hiding**: suppose q is the number of challenge messages sampled by the adversary, the* FHIDE *(Figure 5.6) requires the negligibility of the following advantages:*

$$\mathsf{Adv}^{\mathsf{fhide}}_{\mathcal{A},\mathsf{MIFE}}(\lambda) := \left|\Pr[\mathsf{FHIDE}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda) = 1] - \frac{1}{2}\right| .$$

$\underline{\mathsf{FHIDE}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]} \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{KGEN}_{\mathsf{msk}}(\cdot),\mathrm{ENC}_{\mathsf{msk}}(\cdot)}(1^\lambda)$
for $i \leftarrow 1, q$:
  for $j \leftarrow 1, n$:
    $C^*_{j,i} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, j, M^b_{j,i})$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}^{(\cdot)}_{\mathsf{msk}},\mathrm{ENC}_{\{\mathsf{msk}\}}(\cdot)}\left(1^\lambda, \left\{C^*_{i,j}\right\}_{i\in[q],j\in[n]}\right)$
return $b = b' \wedge \mathrm{Valid}(\mathrm{L}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})$

$\underline{\mathrm{KGEN}_{\mathsf{msk}}(f_0, f_1)}$:
$\mathrm{L} \leftarrow \mathrm{L} \cup \{(f_0, f_1)\}$
return $\mathsf{KGen}(\mathsf{msk}, f_b)$

$\underline{\mathrm{Valid}(\mathrm{L}, \left\{\left(\overrightarrow{M_i^0}, \overrightarrow{M_i^1}\right)\right\}_{i\in[q]})}$:
if $\exists f \in \mathrm{L}, \exists i \in [q] : f_0(\overrightarrow{M}_i^0) \neq f_1(\overrightarrow{M}_i^1)$:
  return 0
return 1

Figure 5.6: The Function-Hiding experiment is defined similarly to IND-MIFE-CPA, with the KGen choosing between one of the two functions the adversary receives. The Valid procedure enforces that for any combination of queried messages, the two functions return the same outputs. Throughout the chapter, we also refer to this notion as *full* IND-MIFE-CPA (note that here *full* is being used in a different context than *adaptive*).

5.2 UBK-Secure Implementations from One-Way MIFE

The main catalyst of this work is the investigation of theoretical means under which *unbreakability* can be achieved. In doing so, we first point out that *indistinguishability* may

be insufficient as a security notion. Imagine the simple case of a MIFE scheme (private-key setting) supporting pseudorandom permutations, introduced graphically in Figure 5.7. The indistinguishability security notion quickly becomes prohibitive by imposing computational restrictions on adversaries: a valid attacker asking for functional keys corresponding to a pseudorandom permutation $\mathcal{E}$, can query the encryption oracle only for tuples $(K^0, M_1^0, M_2^0, \ldots, M_n^0)$ and $(K^1, M_1^1, M_2^1, \ldots, M_n^1)$ such that:

$$\mathcal{E}.\mathsf{Enc}(K^0, M_1^0||M_2^0||\ldots||M_n^0) = \mathcal{E}.\mathsf{Enc}(K^1, M_1^1||M_2^1||\ldots||M_n^1) \ .$$

Since multiple and distinct message components $(M_i^0, M_i^1)_{i\in[n]}$ can be encrypted during the security experiment, then it must be the case that $K^0 = K^1$[2], and by the fact that $\mathcal{E}$ is a keyed permutation we have $M_i^0 = M_i^1$. Put differently, indistinguishability may merely not be the right security notion for modelling the power of a real-world adversary with respect to a class of pseudorandom permutations.

Instead, we remark that our goal of achieving *unbreakability* for implementations of block ciphers shares similarities with the general notion of *one-wayness*. In what follows, we formalize *one-wayness* for MIFE and show it suffices to attain *unbreakability*. Finally, we show a similar result holds in the private-key case, although with a significant increase in the arity of the supported functionality.

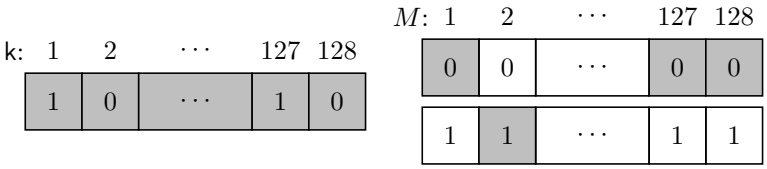

Figure 5.7: Ciphertexts are provided for the bits of the key K, as well as for each of the bits in the binary decomposition of the message. For ease of exposition, we assume both the key and the message are 128-bits long, with $M = (0, 1, \ldots, 0, 0)$.

5.2.1 One-Wayness for MIFE

Assume MIFE is an n-input scheme. The *one-wayness* game is defined with respect to a *challenge* index set $\mathcal{I} \subseteq [n]$ and a function f. The adversary receives the public-keys[3] from $\overline{\mathcal{I}} \coloneqq [n] - \mathcal{I}$, denoted $\{\mathsf{mpk}_i\}_{i\in\overline{\mathcal{I}}}$, and a set of challenge ciphertexts corresponding to $\mathcal{I}$, and written $\{C_i^*\}_{i\in\mathcal{I}}$. The challenge ciphertext(s) are build by the one-wayness security experiment, which samples uniformly at random a set of plaintexts $(M_1, \ldots, M_n)$, encrypts them, and provides (part of) them to the adversary as the challenge(s). The winning condition says the adversary wins if it successfully recovers at least one of the plaintexts corresponding to the ciphertexts it received.

Definition 5.5 (One-Wayness for MIFE)**.** *Let* MIFE *be a multi-input functional encryption scheme in the public-key (private-key) setting. The advantage of any* PPT *adversary* $\mathcal{A}$ *against the* ***one-wayness*** *of* MIFE *with respect to a function* $f : \mathcal{M}_1 \times \ldots \times \mathcal{M}_n \to \mathcal{C}$ *and an index set* $\mathcal{I}$*, is defined as:*

$$\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{A},\mathsf{MIFE}}(\lambda) \coloneqq \Pr[\mathsf{OW\text{-}MIFE}^{\mathcal{A},f,\mathcal{I}}_{\mathsf{MIFE}}(\lambda) = 1] \ ,$$

[2]Otherwise, the adversary trivially wins the game.

[3]If in the private-key setting, access to encryption oracles is given.

where the security experiment $\mathsf{OW\text{-}MIFE}^{\mathcal{A}}_{\mathsf{MIFE}}(\lambda)$ *is defined in Figure 5.8 for the public (private) setting. We say* MIFE *is* OW-MIFE*-secure with respect to* f *and* $\mathcal{I}$ *if* $\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{A},\mathsf{MIFE}}(\lambda)$ *is negligible.*

$\mathsf{OW\text{-}MIFE}^{\mathcal{A},f,\mathcal{I}}_{\mathsf{MIFE}}(\lambda)$: // Public-Key FE	$\mathsf{OW\text{-}MIFE}^{\mathcal{A},f,\mathcal{I}}_{\mathsf{MIFE}}(\lambda)$: // Private-Key FE
$(\mathsf{msk}, \{\mathsf{mpk}_1, \ldots, \mathsf{mpk}_n\}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$	$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{MIFE.KGen}(\mathsf{msk}, f)$	$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{MIFE.KGen}(\mathsf{msk}, f)$
$(M_1, \ldots, M_n) \leftarrow\!\!\$\ \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$	$(M_1, \ldots, M_n) \leftarrow\!\!\$\ \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$
$C^*_i \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{mpk}_i, M_i), \forall i \in \mathcal{I}$	$C^*_i \leftarrow\!\!\$\ \mathsf{MIFE.Enc}(\mathsf{msk}, M_i), \forall i \in \mathcal{I}$
$\{N_i\}_{i\in\mathcal{I}} \leftarrow\!\!\$\ \mathcal{A}(\mathsf{sk}_f, \{\mathsf{mpk}_i\}_{i\in\overline{\mathcal{I}}}, \{C^*_i\}_{i\in\mathcal{I}})$	$\{N_i\}_{i\in\mathcal{I}} \leftarrow\!\!\$\ \mathcal{A}^{\mathrm{ENC}_{\mathsf{msk},\overline{\mathcal{I}}}(\cdot)}(\mathsf{sk}_f, \{C^*_i\}_{i\in\mathcal{I}})$
return $\bigvee_{i\in\mathcal{I}} (M_i = N_i)$	**return** $\bigvee_{i\in\mathcal{I}} (M_i = N_i)$

Figure 5.8: The one-wayness security experiments defined for functional encryption schemes in the public (left) and private (right) settings. In both cases, the adversary is provided with ciphertexts corresponding to randomly sampled messages and is asked to "extract" the underlying inputs. The adversary wins if it can successfully recover at least one of the inputs.

The primary expectation is hardness in recovering any of the encrypted plaintexts if functional-keys are issued for *one-way functions* candidates f.

5.2.2 OW-MIFE ⇒ UBK

As stated in the introductory part, a major goal is to study to what extent a UBK implementation of a pre-specified block-cipher $\mathcal{E}$ is achievable. We show a two-input functional encryption scheme being one-way secure concerning a class of pseudorandom permutations, is sufficient in achieving UBK-secure implementations for that class.

Lemma 5.1 (OW-MIFE ⇒ UBK)**.** *Let* $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ *denote a secure pseudorandom permutation (block-cipher). Let* MIFE *denote a two-input public-key functional encryption scheme achieving* OW-MIFE *security with respect to* $\mathcal{E}$ *and index set* $\mathcal{I} = \{1\}$*. The program* $\mathsf{C}_{\mathcal{E}}.\mathsf{Eval}$ *described in Figure 5.9, achieves* UBK*-security (Definition 5.2) against any* PPT *adversary* $\mathcal{A}$ *under the following advantage:*

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_{\mathcal{E}}.\mathsf{Eval}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 5.1. We take the contrapositive. Let $\mathcal{A}$ be a PPT adversary against the UBK-security of $\mathsf{C.Eval}^r_k$. We construct a PPT reduction $\mathcal{R}$ that acts as an adversary against the OW-MIFE game while producing an implementation for $\mathcal{A}$ as follows: first, the OW-MIFE game samples the pair of keys $(\mathsf{msk}, \{\mathsf{mpk}_1, \mathsf{mpk}_2\}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$. Next, $\mathcal{R}$ is given the functional key $\mathsf{sk}_{\mathcal{E}}$ corresponding to $\mathcal{E}$. Since the index set $\mathcal{I} \leftarrow \{1\}$, the OW-MIFE challenger replies by providing mpk_2 (since $1 \in \mathcal{I}$ and mpk_1 is not revealed) and the challenge ciphertext corresponding to mpk_1 (that is $C_K = \mathsf{Enc}(\mathsf{mpk}_1, K)$). $\mathcal{R}$ now defines the implementation $\mathsf{C.Eval}^r_k$ to include the challenge ciphertext and parameters: $(C_K, \mathsf{mpk}_2, \mathsf{sk}_{\mathcal{E}})$. Note that using these values, the UBK adversary $\mathcal{A}$ can obtain the encryption of any message $M' \in \mathcal{M}$.

Assume $\mathcal{A}$ returns the key K' after interacting with the implementation. The reduction $\mathcal{R}$ simply returns K' as its output. If $K = K'$ with non-negligible probability, then $\mathcal{R}$ succeeds

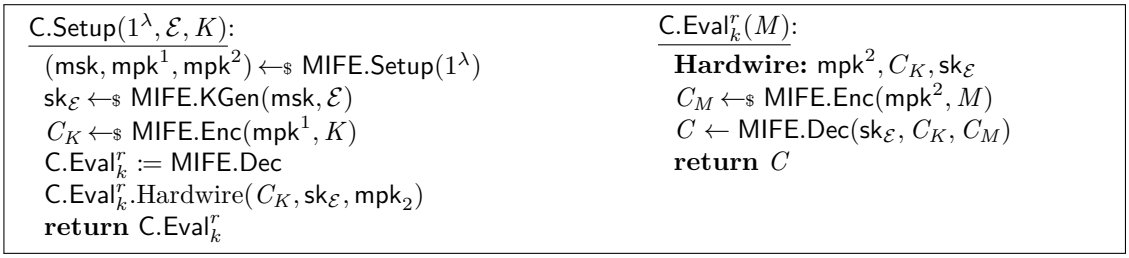

Figure 5.9: The candidate construction for obtaining a UBK-secure implementation of $\mathcal{E}$, given a OW-MIFE secure 2-input functional encryption scheme MIFE. Correctness follows immediately from the correctness of the MIFE scheme.

in breaking the OW-MIFE game, with the same probability that $\mathcal{A}$ breaks UBK, therefore having: $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}.\mathsf{Eval}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda)$. □

5.2.3 The Private-Key Setting

The same result holds with respect to private-key MIFEs, but in a *significantly* different manner. First, we remark that we cannot use a 2-input scheme in the same way we did in the public-key setting: there is no master public key to be used to encrypt the message M. Thus, instead of trying to substitute the role of mpk_2, we simulate the entire message space by providing the encryptions of $\{0,1\}$ for the entire binary length of M. Thus, the compiled version selects the correct encodings of 0s and 1s based on the binary decomposition of M and computes the correct input for the decryption algorithm. Knowing this information, one can decrypt and learn $\mathcal{E}(K, M)$. If the MIFE scheme is function hiding, then K may be "embedded" in the functional key. However, for the general case, we choose to put it alongside the encodings 0s and 1s.

Lemma 5.2. *Let $\mathcal{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ denote a secure pseudorandom permutation (block cipher) where $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \{0,1\}^n$. Let* MIFE *denote a private-key, $(n+1)$-input functional encryption scheme that is* OW-MIFE*-secure w.r.t. $\mathcal{E}$ and index set $\mathcal{I} = \{n+1\}$. The program in Figure 5.10 implements $\mathcal{E}$ and achieves* UBK*-security, against any* PPT *adversary $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}.\mathsf{Eval}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 5.2. The correctness comes straightforward. UBK-security follows immediately. Let $\mathcal{A}$ be the UBK adversary. We construct $\mathcal{R}$ that simulates the UBK security experiment (in the view of $\mathcal{A}$), while modelling a OW-MIFE adversary for the one-wayness game. Thus, the OW-MIFE game samples uniformly at random a message $M = (M_1, \ldots, M_n, M_{n+1}{=}K)$ and provides $\mathcal{R}$ with C^*_{n+1} and the functional key corresponding to the $\mathcal{E}$.Enc. Next, since $\mathcal{I} = \{n+1\}$, $\mathcal{R}$ can query the $\mathsf{Enc}_{\mathsf{msk},[n]}(\cdot)$ oracles for the encodings of 0s and 1s (for position in $[n]$), such that it can simulate $\{C_{i,b}\})_{i\in[n],b\in\{0,1\}}$. In what follows, $\mathcal{R}$ will set C_K to be C^*_{n+1}, as received from the challenger and send the implementation to $\mathcal{A}$. Note that $\mathcal{R}$ emulates the UBK setup in the view of $\mathcal{A}$. Assume that $\mathcal{A}$ returns K with $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}}(\lambda)$. Knowing K, $\mathcal{R}$ returns it as its guess for the challenge ciphertext (corresponding to $n{+}1^{\text{th}}$ input) and wins the OW-MIFE game with the same advantage $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}.\mathsf{Eval}}(\lambda)$. □

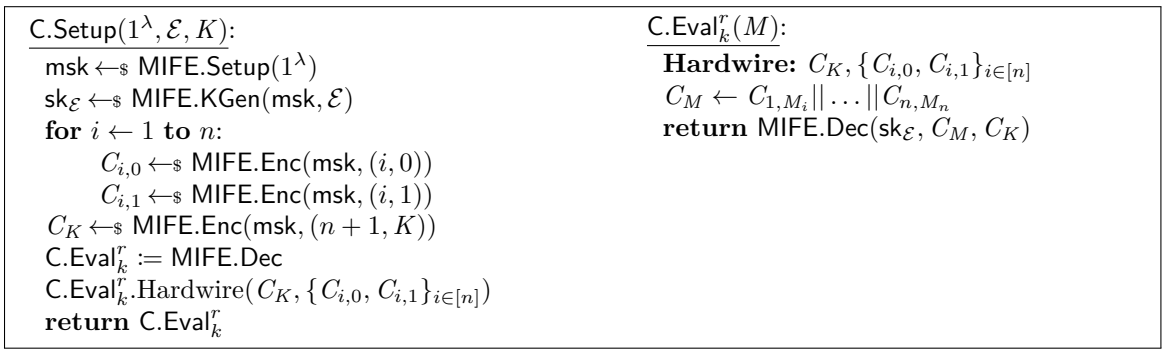

Figure 5.10: A compiler providing a UBK-secure implementation for a given block cipher $\mathcal{E}$ and a key K. The construction uses a n-input OW-MIFE-secure functional encryption scheme MIFE (private key setting).

Weakening the Definition. As one can easily observe from the reduction in the proof of Lemma 5.2, there is no real reason to provide the reduction $\mathcal{R}$ with oracle access to the encryption procedure. Crafting a UBK-secure implementation for $f : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$, requires only the encryption of the key of f and the encodings of 0s and 1s. It is easy to show such a security notion is implied by the OW-MIFE notion we provide in Figure 5.8, and it implies UBK-security.

5.3 Achieving One-Wayness from Standard Assumptions

In this part, we introduce a simple construction of a MIFE scheme that enjoys one-wayness. It uses as an underlying primitive the private-key MIFE transform from [BKS16]; the latter relies on any function-hiding private-key FE scheme, which can be instantiated from standard assumptions. We prove the transform preserves one-wayness and show that it can be based on an FE scheme achieving one-wayness by looking into the seminal FE construction for general circuits by Goldwasser *et al.* [GKP+13].

5.3.1 A One-Way MIFE Transform in the Private-Key Setting

Roadmap. The bulk of this section follows from combining a number of steps: (1) first, we review the construction of Goldwasser *et al.* in [GKP+13], and point out that in spite of its complexity, it will serve as our starting point in achieving a candidate OW-MIFE scheme; (2) we then obtain a candidate construction in the private-key setting by simply *not publishing* the master public key; (3) by using the result of Brakerski and Segev [BS15], we obtain, in a generic manner, a transform that achieves *function-hiding* in the private-key setting; (4) finally, in order to obtain an n-input MIFE scheme in the private-key setting, we have the choice to either apply the BKS generic transform in [BKS16], which increases the arity of the function by 1 for each step, or, to apply the transform in [KS17], which doubles the arity of the supported function at each step, at the cost of supporting "larger" circuits. Our goal is a final construction achieving one-wayness. To this end, we transit through a chain of implications, with each step essentially relying on the one-wayness of the underlying scheme. Proofs are given for *one-wayness* of the underlying FE construction, as well as for each generic transform to be used.

$\underline{\mathsf{FE.Setup}(1^\lambda, \ell, n)}$:
$\mathsf{msk} \leftarrow \emptyset$
$\mathsf{mpk} \leftarrow \emptyset$
for $i \leftarrow 1$ **to** ℓ:
$(\mathsf{mpk}_i, \mathsf{msk}_i) \leftarrow\!\!\$\ \mathsf{ABE}_2.\mathsf{Setup}(1^\lambda)$
$\mathsf{mpk} \leftarrow \mathsf{mpk} \cup \mathsf{mpk}_i$
$\mathsf{msk} \leftarrow \mathsf{msk} \cup \mathsf{msk}_i$
return $(\mathsf{msk}, \mathsf{mpk})$

$\underline{\mathsf{FE.KGen}(\mathsf{msk}, f)}$:
$\mathsf{sk}_f \leftarrow \emptyset$
for $i \leftarrow 1$ **to** ℓ:
$\mathsf{sk}_i \leftarrow\!\!\$\ \mathsf{ABE}_2.\mathsf{KGen}(\mathsf{msk}_i, \mathsf{FHE.Eval}_f^i)$
$\mathsf{sk}_f \leftarrow \mathsf{sk}_f \cup \mathsf{sk}_i$
return sk_f

$\underline{\mathsf{FE.Enc}(\mathsf{mpk}, M)}$:
$(\mathsf{hpk}, \mathsf{hsk}) \leftarrow\!\!\$\ \mathsf{FHE.Setup}(1^\lambda)$
for $i \leftarrow 1$ **to** n:
$\phi_i \leftarrow\!\!\$\ \mathsf{FHE.Enc}(\mathsf{hpk}, M_i)$
$\Phi \leftarrow (\phi_1, \ldots, \phi_n)$
$(\Gamma, L_1^0, L_1^1, \ldots, L_\ell^0, L_\ell^1) \leftarrow\!\!\$$
$\leftarrow\!\!\$\ \mathsf{GS.Garble}(\mathsf{FHE.Dec}(\mathsf{hsk}, \cdot))$
for $i \leftarrow 1$ **to** ℓ:
$c_i \leftarrow\!\!\$\ \mathsf{ABE}_2.\mathsf{Enc}(\mathsf{mpk}_i, (\mathsf{hpk}, \Phi), L_i^0, L_i^1)$
$C \leftarrow (\Gamma, c_1, \ldots, c_\ell)$
return C

$\underline{\mathsf{FE.Dec}(\mathsf{sk}_f, C)}$:
$(\Gamma, c_1, \ldots, c_\ell) \leftarrow C$
for $i \leftarrow 1$ **to** ℓ:
$L_i^{d_i} \leftarrow \mathsf{ABE}_2.\mathsf{Dec}(\mathsf{sk}_i, c_i)$
return $\mathsf{GS.Eval}(\Gamma, L_1^{d_1}, \ldots, L_\ell^{d_\ell})$

Figure 5.11: The functional encryption scheme for boolean circuits $\mathcal{C}_f : \{0,1\}^n \to \{0,1\}$ as introduced in [GKP+13]. ℓ stands for the ciphertext's lenght of FHE, while $\mathsf{FHE.Eval}_f^i : \mathcal{K} \times \{0,1\}^{n \cdot \ell} \to \{0,1\}$ denotes the function that applies $\mathcal{C}_f$ on the encrypted input.

5.3.1.1 Step 1 - FE for Boolean Circuits [GKP+13].

Special classes of functions, such as Boolean circuits with 1-bit of output, are functional-encryption suitable. Such FE constructions can be achieved assuming the existence of ABE (Definition 2.9), FHE (Definition 2.6) and of Garbling Schemes (Definition 2.7).

The construction in [GKP+13]. Goldwasser *et al.* propose to regard FE for circuits with a single-bit of output through the lenses of FHE (Figure 5.11). In their scheme, the *encryption procedure*:

- Generates on the fly the keys for an FHE scheme – namely $(\mathsf{hpk}, \mathsf{hsk})$ – and encrypts the input M bitwise; let Φ denote the FHE ciphertext.
- Next, Yao's garbling scheme GS is used to garble the circuit "$\mathsf{FHE.Dec}(\mathsf{hsk}, \cdot)$" and obtain two labels L_i^0, L_i^1 per input bit M_i;
- Finally, the scheme encrypts Φ w.r.t. multiple ABE's schemes. In some sense, Φ corresponds to an attribute: if $\mathcal{C}_f(M_1, \ldots, M_n) = 1$ a label L^0 is revealed. Otherwise, a label L^1 is returned (this is obtained via a two-outcome ABE).

A *functional key* for a circuit consists in an ABE key for the "FHE.Eval" circuit. The intuition is that one decrypts an ABE ciphertext with an ABE key; this corresponds to applying FHE.Eval over a FHE ciphertext. Depending on the output (which is a bit b), a label L_i^b is revealed. Once the labels are known and provided to the garbled circuit (as part of the ciphertext), the decryptor evaluates and obtains $\mathsf{FHE.Dec}(f(\Phi))$, thus yielding the

expected output in a functional manner. Thus, the master keys for the FE scheme consist only of ABEs' msk and mpk. The number of ABE keys needed corresponds to the length of the FHE ciphertext.

As suggested by the authors, the result can be extended for a circuit with a constant number n of output bits almost trivially, by "replicating" the construction in [GKP+13] for each bit of output, incurring a factor n blow-up in the ciphertext length. Most importantly, the scheme is proven semantic secure: $\Pr[\mathsf{FULL\text{-}SIM\text{-}FE}^{\mathcal{A},\mathcal{S}}_{\mathsf{FE}}(\lambda) = 1] - \frac{1}{2}$ is negligible, where the FULL-SIM-FE experiment is described in Figure 2.3.

Lemma 5.3. *Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a secure pseudorandom permutation. Let* FE *be the* FULL-SIM-FE*-secure (public-key) functional encryption scheme introduced in [GKP+13] supporting circuits $\mathcal{C}_f : \{0,1\}^{k+n} \to \{0,1\}^n$. The* FE *scheme enjoys one-wayness (Definition 5.5) w.r.t. f and index set $\mathcal{I} = \{1\}$.*

Lemma 5.3. Informally, FULL-SIM-FE guarantees that the decryptor does not learn more information on M than what $f(M)$ reveals. Let us suppose that an adversary $\mathcal{A}$ against the one-wayness of the FE scheme exists; we then build a PPT algorithm R that runs $\mathcal{A}$ and wins the FULL-SIM-FE game. Assuming the existence of a simulator $\mathcal{S}$, the FULL-SIM-FE game proceeds by sampling $(\mathsf{msk}, \mathsf{mpk})$, and then receiving (M, f) from R. Depending on the setting, the challenger replies with sk_f and a ciphertext C which is either correctly generated or is obtained from $\mathcal{S}$. R forwards the ciphertext to $\mathcal{A}$. If $\mathcal{A}$ replies with M, then R returns $b' = 0$. If $\mathcal{A}$ replies with any other value $M' \neq M$, then R returns $b' = 1$.

Analysis. If $b = 0$, then $\mathcal{A}$ returns M with $\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{A},\mathsf{FE}}(\lambda)$, as it simulates perfectly the setting of the OW-MIFE game. On the other hand, when $b = 1$, then $\mathcal{A}$ receives a ciphertext that leaks M only through $f(M)$, $\mathcal{A}$'s probability of returning M is essentially bounded by $\mathsf{Adv}^{\mathsf{owp}}_{\mathcal{A},f}(\lambda)$. With overwhelming probability, for this second case, the adversary will return $M' \neq M$. Directly, R returns 1 with probability $1 - 1/2 \cdot \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{A,\mathsf{FE}}(\lambda) - 1/2 \cdot \mathsf{Adv}^{\mathsf{owp}}_{A,f}(\lambda)$. Now, if $\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{A,\mathsf{FE}}(\lambda) \notin \textsc{Negl}$ then R breaks FULL-SIM-FE. □

5.3.1.2 Step 2 - FE in the Private Key Setting.

Given any indistinguishable secure functional encryption scheme in the public-key setting, its counterpart in the private-key setting is also secure. This observation is straightforward, as remarked by [BS15, p. 9], and we show it holds even when one-wayness is considered, as opposed to indistinguishability Put differently, Figure 5.11 immediately yields a secure one-way scheme in the private-key setting for circuits with one bit of output.

Lemma 5.4. *Let $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be a secure pseudorandom permutation. Let* FE *be a public-key functional encryption scheme supporting circuits $\mathcal{C}_f : \{0,1\}^{k+n} \to \{0,1\}^n$, and let* $\overline{\mathsf{FE}}$ *be the private-key scheme obtained by setting the master public-key as part of the master secret key. If* FE *is* OW-MIFE*-secure w.r.t. f and index set $\mathcal{I}$, then* $\overline{\mathsf{FE}}$ *enjoys* OW-MIFE *security w.r.t. the same f and $\mathcal{I}$.*

Proof. The reduction is straightforward. $\mathcal{A}$, the OW-MIFE adversary against $\overline{\mathsf{FE}}$, receives from the reduction $\mathcal{R}$ a ciphertext and a functional key corresponding to f, generated by the OW-MIFE experiment defined for FE. The challenge ciphertext C^* corresponds to M^*. We note that as per our definition of OW-MIFE, the adversary cannot make any encryption request as it already has the challenge ciphertext corresponding to the sole input of the

function – although a stronger one-way definition may allow for this. Hence, if $\mathcal{A}$ returns M^*, then $\mathcal{R}$ wins the game with the same advantage by simply returning M^*. Therefore, $\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{A},\overline{\mathsf{FE}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{FE}}(\lambda)$. □

5.3.1.3 Step 3 - Achieving Function Hiding.

Intuitively, function-hiding ensures that a public-key encryption scheme does not leak information about the function through the key it generates. Brakerski and Segev [BS15] show a simple transform, applicable generically to any private-key FE scheme, that results in a scheme achieving *function hiding*. We give the original transform as it is, but note that the double encryption technique [NY90] plays no role in the context of *one-wayness*.

Definition 5.6 ([BS15])**.** *Let* FE *be a private-key* FE *scheme. A function-hiding private-key functional encryption scheme* $\overline{\mathsf{FE}}$ *is obtained as follows:*

- $\overline{\mathsf{FE}}.\mathsf{Setup}(1^\lambda)$: *samples* $\mathsf{msk} \leftarrow_\$ \mathsf{FE.Setup}\ (1^\lambda)$ *and two secret-key* $(K, K') \leftarrow_\$ \mathsf{SE.Setup}^2(1^\lambda)$ *from a semantic-secure secret-key encryption scheme* SE*. It sets* $\overline{\mathsf{msk}} \leftarrow (\mathsf{msk}, K, K')$.

- $\overline{\mathsf{FE}}.\mathsf{Enc}(\mathsf{msk}, M)$: *given a plaintext* M*, the ciphertext is obtained by running the underlying scheme as follows:* $C \leftarrow_\$ \mathsf{FE.Enc}(\mathsf{msk}, (M, \bot, K, \bot))$.

- $\overline{\mathsf{FE}}.\mathsf{KGen}(\mathsf{msk}, f)$: *the key for a function* f *is generated as follows - first the circuit describing* f *is encrypted w.r.t.* K, K' *and obtaining* c, c'*. A circuit* $U_{c,c'}$ *which decrypts* c *and applies* f *on input is constructed. Then* $\mathsf{sk}_f \leftarrow_\$ \mathsf{FE.KGen}(\mathsf{msk}, \mathcal{C}_{U_{c,c'}})$ *is returned.*

- $\overline{\mathsf{FE}}.\mathsf{Dec}(\mathsf{sk}_f, C)$: *applying* $U_{c,c'}$ *on* $(M, \bot, K, \bot)$ *is equivalent to the application of* $f(M)$.

An important remark on the construction above is the fact that K gets encrypted, but is not part of the message space. It is, in turn, part of the master secret key. Thus, we assume that whatever functional keys are queried, they will only process the message-dependent part in the given plaintext.

Lemma 5.5. *Let* $f : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a secure pseudorandom permutation and let* FE *be a functional encryption scheme supporting circuits* $\mathcal{C}_f : \{0,1\}^{2\cdot(k+n)+2\cdot k} \to \{0,1\}^n$*. If* FE *is a* OW-MIFE*-secure private-key functional encryption scheme w.r.t.* f *and index set* $\mathcal{I} = \{1\}$*, then, the* $\overline{\mathsf{FE}}$ *in Definition 5.6 enjoys* OW-MIFE*-security for the same function* f *and index set* $\mathcal{I}$*.*

Lemma 5.5. Assuming that $\mathcal{A}$ wins the OW-MIFE game against $\overline{\mathsf{FE}}$, we build a reduction $\mathcal{R}$ winning the OW-MIFE game against the underlying scheme FE. First, the reduction $\mathcal{R}$ defines the message space of the form $(M, \bot, K, \bot)$. Next, $\mathcal{R}$ samples (K, K'), and computes the corresponding $U_{c,c'}$ for f. We note that (K, K') are not formally part of the message, as they are included in $\overline{\mathsf{msk}}$. By interacting with the OW-MIFE challenger, a functional key for $U_{c,c'}$ and a challenge ciphertext C^* corresponding to $(M, \bot, K, \bot)$ are obtained by $\mathcal{R}$. $\mathcal{R}$ forwards them to $\mathcal{A}$. When $\mathcal{A}$ returns its "guess" for M', $\mathcal{R}$ forwards it to the challenger and wins the OW-MIFE game against FE under the same advantage that $\mathcal{A}$ wins the OW-MIFE game against $\overline{\mathsf{FE}}$. □

5.3.1.4 Step 4 - Achieving a Multi-Input Scheme.

Expanding a single-input FE scheme (private-key setting) into one supporting multiple inputs can be done via a generic transform. The idea is to split a single input of – say $n \cdot l$ – into n inputs of length l, while having an aggregator that "glues" the inputs before a function is to be applied on them. Two main transforms have been proposed, working sequentially [BKS16] *or* using a "divide-and-conquer" approach [KS17]. Here we focus on the straightforward, sequential approach, which relies on the following building blocks: (1) a function-hiding, private-key single-input scheme FE_1; (2) a private-key t-input scheme FE_t; (3) a (puncturable) pseudorandom function PRF.

We review the transform in [BKS16] – from now on referred to as BKS – which works as follows:

- $\mathsf{MIFE.Setup}$: samples $\mathsf{msk}_{out} \leftarrow\!\!\$\ \mathsf{FE}_1.\mathsf{Setup}(1^\lambda)$ and $\mathsf{msk}_{in} \leftarrow\!\!\$\ \mathsf{FE}_t.\mathsf{Setup}(1^\lambda)$. The master key is set as $\mathsf{msk} \leftarrow (\mathsf{msk}_{in}, \mathsf{msk}_{out})$.

- $\mathsf{MIFE.KGen}$: given msk and $f \in \mathcal{F}_\lambda$ and z - a randomly sampled bitstring, a functional key for $\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{FE}_1.\mathsf{KGen}(\mathsf{msk}_{out}, D_{f,\perp,z,\perp})$ is provided:

$D_{f_0,f_1,z,u}(\mathsf{msk}^*, K, w)$:	$\mathcal{C}_{f_w}((x_1,x_2),x_3,\ldots x_t,x_{t+1})$:
if $\mathsf{msk}^* = \perp$ **return** u $r \leftarrow \mathsf{PRF.Eval}(K, z)$ **return** $\mathsf{FE}_t.\mathsf{KGen}(\mathsf{msk}^*, \mathcal{C}_{f_w}; r)$	**return** $f(x_1, x_2, x_3, \ldots x_{t+1})$

- $\mathsf{MIFE.Enc}$: given msk, the message $M = x_i$ and index position i, the encryption proceeds as follows:

  – for $(x_1, i = 1)$: a new $\mathsf{msk}^* \leftarrow\!\!\$\ \mathsf{FE}_t(1^\lambda)$ is generated on the fly, as well as a PRF key $K \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$ and $s \leftarrow\!\!\$\ \{0,1\}^\lambda$. Then, the following are computed:

$$C_1 \leftarrow\!\!\$\ \mathsf{FE}_1.\mathsf{Enc}(\mathsf{msk}_{out}, (\mathsf{msk}^*, K, 0))$$

$$\mathsf{sk}_1 \leftarrow\!\!\$\ \mathsf{FE}_t.\mathsf{KGen}(\mathsf{msk}_{in}, AGG_{x_1,\perp,0,s,\mathsf{msk}^*,K})$$

  where AGG is defined as follows:

$AGG_{x_1^0,x_1^1,a,s,\mathsf{msk}^*,K}((x_2^0,x_2^1,\tau_2,s_2,v_2),\ldots,(x_{t+1}^0,x_{t+1}^1,\tau_{t+1},s_{t+1},v_{t+1}))$:

if $s_2 = \ldots = s_t = s$:
  return $(v_2, \ldots, v_t)$ and HALT
Set $x_i \leftarrow x_i^a$ for $i \leftarrow 2$ **to** $t+1$
Set $r_i \leftarrow \mathsf{PRF.Eval}(K, \tau_i)$ for $i \leftarrow 2$ **to** $t+1$
return $\Big(\mathsf{FE}_t.\mathsf{Enc}(\mathsf{msk}^*, (x_1, x_2), 1; r_1), \ldots, \mathsf{FE}_t.\mathsf{Enc}(\mathsf{msk}^*, x_{t+1}, t; r_t)\Big)$

  – for $(x_i, i \in \{2, \ldots, t+1\})$:

$$C_i \leftarrow\!\!\$\ \mathsf{FE}_t.\mathsf{Enc}(\mathsf{msk}_{in}, (x_i, \perp, \tau_i, \perp, \perp), i-1)$$

- $\mathsf{MIFE.Dec}$: first, a secret sk^* is obtained by decrypting C_1 under sk_f. Then a ciphertext C^* is obtained by decrypting $(C_2, \ldots, C_{t+1})$ under sk_1. Finally, C^* is decrypted under sk^* and the $f(x_1, \ldots, x_t, x_{t+1})$ is recovered.

Remark 5.1. *As shown by its authors, the BKS transform described above enjoys the full selective security notion* s-IND-MIFE-CPA *(i.e.* FHIDE*). The authors also show how to reach adaptive security, but this does not constitute a priority for this work. As an independent remark with potentially useful consequences, we are now able to state that the original security property achieved by the scheme suffices for proving directly that puncturable pseudorandom functions admit* UBK*-secure implementations.*

Lemma 5.6 (UBK Implementations for Puncturable PRFs)**.** *Let* MIFE *be the $n{+}1$ input and fully* s-IND-FE-CPA*-secure scheme defined over $\{0,1\} \times \{0,1\} \times \ldots \{0,1\} \times \mathcal{K}$ and supporting functional keys for a class $\mathfrak{C}$ of puncturable* PRF*s (Definition 2.2). Let* pPRF $=$ (Setup, Eval, Puncture) *denote a puncturable pseudorandom function such that* Eval $: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$*. Let* MIFE *denote a* s-IND-FE-CPA*-secure $n{+}1$ input functional encryption scheme for general circuits. Then, an* UBK *secure implementation of* pPRF *exists such that:*

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C.Eval}^r_k}(\lambda) \leq \mathsf{Adv}^{\mathsf{fhide}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 5.6. The description of the implementation is trivial: it consists of the encryptions of 0 and 1 for each of the first n input slots, as well as the functional key that computes pPRF (analogous to the one presented in Figure 5.10).

The implementation is UBK secure down to the full s-IND-MIFE-CPA security of the underlying MIFE scheme. The reduction algorithm $\mathcal{R}$ sets as the challenge messages n pairs of the form $(0,0)$ and $(1,1)$ (for each input slot). For the final slot – corresponding to the key – two queries are made: on the one hand side, for the *normal* pPRF key K_{pPRF} and on the other side, for the punctured key K^*_{pPRF} together with real value at the punctured point; next, two functional key queries are made, one for the circuit computing the pPRF under the *normal* key, and one computing the pPRF under the punctured key and also using the real value pPRF evaluation at the punctured point.

By the correctness of the pPRF, the two circuits are equivalent. Moreover, the two circuits return the same value when both are fed with any input M. The full s-IND-FE-CPA game returns the MIFE encryptions of 0 and 1 per each position $i \in [n]$, the encoding of either the normal or the punctured key for position $n{+}1$, as well as one of the two functional keys. If an UBK adversary succeeds in recovering the (normal or punctured) key, the reduction chooses to return 0 and 1 depending on a normal/punctured key and wins the full s-IND-FE-CPA game subsequently. □

We state two immediate consequences with more relevant impact to practice:

Proposition 5.1 (A UBK-secure Stream Cipher Implementation)**.** *Let* pPRF $: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *denote a puncturable pseudorandom functions, and let* $\mathsf{C.Eval}^r_k$ *be its* UBK *secure implementation. Let* SE *denote a stream cipher where* SE.Setup $:=$ pPRF.Setup *and* SE.Enc $:=$ (pPRF.Eval$(K,R) \oplus M, R$) *Then, an* UBK*-secure implementation for* SE *exists.*

Proposition 5.1. The implementation queries the $\mathsf{C.Eval}^r_k$ with randomness R [4] and obtains pPRF.Eval(K,R). The proof follows immediately via the security of the implementation of the pPRF. □

[4] Which can be generated by querying another UBK-secure implementation of pPRF′ with message M.

A similar result can be stated with respect to block ciphers. If there exists a UBK-secure implementation for a pPRF, the Luby-Rackoff [LR86] transform provides immediately a block cipher.

Proposition 5.2 (A UBK-secure Block Cipher Implementation)**.** *Let* $\mathsf{pPRF} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *denote a puncturable pseudorandom functions, and let* $\mathsf{C.Eval}_k^r$ *be its* UBK *secure implementation. Let* $\mathsf{PRP} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *denote the pseudorandom permutation obtained using the Luby-Rackoff transformation. Then, a* UBK*-secure implementation for* PRP *exists.*

Proposition 5.2. Any adversary extracting the key can be converted into an adversary winning the UBK game against the $\mathsf{C.Eval}_k^r$. □

We now turn to the construction of a UBK-secure implementation for pseudorandom permutations starting from the multi-input functional encryption scheme. There are two main paths one can explore. Essentially, the first one would rely on the fact that the original, single-input, private-key and function-hiding scheme is *one-way* and the transform above produces an implementation by issuing encodings of 0 and 1 as well as a functional key that embeds the key of the permutation. However, such an approach is extremely convoluted as it has to explore the intricate nature of the MIFE scheme by making use of the previous steps described[5]. Thus we defer it to future work.

The second approach is, by far, simpler to follow, as it exploits the already proven full s-IND-MIFE-CPA security of the scheme. Concretely, in the first part, we consider a keyed PRP operating on a short input, say m out of n bits. Equivalently, it has n-m bits of input "fixed" to some constant and m bits "free". We show that any s-IND-MIFE-CPA-secure scheme for this class of PRPs is also OW-MIFE-secure. Then once we showed that such implementations exist, we can show that there exists OW-MIFE-schemes for PRPs with $m{+}1$ free bits, and via n-m transitions, up to n bits of freedom.

Lemma 5.7. *Let* $\mathsf{PRP} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a secure pseudorandom permutation. Let* $\mathsf{PRP}_{K,n-m} : \{0,1\}^m \to \{0,1\}^n$ *denote the restriction of the* PRP *keyed with a randomly sampled* K *and having* m *bits free and* $n-m$ *bits fixed to some constant. Let* MIFE *be a fully* s-IND-MIFE-CPA*-secure scheme with* $m{+}1$ *inputs defined over the message space* $\{0,1\} \times \ldots \times \{0,1\} \times \mathcal{K} \to \{0,1\}^n$ *supporting the circuit representation of* $\mathsf{PRP}_{K,n-m}$*. Then,* MIFE *is* OW-MIFE*-secure with respect to* $\mathsf{PRP}_{K,n-m}$ *and index set* $\{m+1\}$*.*

Proof. Suppose that on the one hand there exists a lookup-table implementation[6] for the restriction $\mathsf{PRP}_{K,n-m}$, containing 2^m entries. On the other hand, suppose there exists a genuine circuit that emulates $\mathsf{PRP}_{K,n-m}$. Both can be build by a reduction $\mathcal{R}$ by having knowledge of the key K.

We proceed via a reduction to the full s-IND-MIFE-CPA game. Let $\mathcal{A}$ stand for the adversary against the OW-MIFE game and $\mathcal{R}$ stand for the reduction. $\mathcal{R}$ sets as the challenge messages for the first n positions the following values: $(\{(0,0),(1,1)\}_{i\in[n]})$. This is to ensure it gets encryptions of 0 and 1 per each position. The full s-IND-MIFE-CPA game replies with $\mathsf{MIFE.Enc}(\mathsf{msk}, 0, i)$ and $\mathsf{MIFE.Enc}(\mathsf{msk}, 1, i)$, which can be used by $\mathcal{R}$ to construct the encryption oracle for the OW-MIFE adversary.

[5] More specifically, of the one-wayness of the single-input FE scheme keyed by msk_{out}.
[6] We assume that the input-output behaviour does not leak the key of the PRP.

For position $n{+}1$, $\mathcal{R}$ prepares the following challenge message: $(K, \$)$, where $\$$ denotes an element sampled uniformly at random over $\mathcal{K}$.

$\mathcal{R}$ also submits two functional-key queries for circuits that encode, on one hand the circuit computing $\mathsf{PRP}_{K,n-m(\cdot)}$ and on the other hand the $\mathsf{LookupTable.Return()}$ functionality.

Receiving all the components from the full s-IND-MIFE-CPA game means that $\mathcal{R}$ receives the functional key for the circuit computing the restriction of the PRP to m bits of freedom (i.e. $\mathsf{PRP}_{K,n-m}(\cdot)$), as well as the encodings of 0/1 per each input and the encoding of $K/\$$ for the $n{+}1^{th}$ input. This suffices to simulate the OW-MIFE game in the view of $\mathcal{A}$.

The simulation is correct, as the two functions are equivalent and constitute valid implementations in the view of $\mathcal{A}$. If the adversary returns the valid key K, $\mathcal{R}$ returns $b' = 0$. When $\mathcal{A}$ returns $K' \neq K$, $\mathcal{R}$ returns a guess $b' \leftarrow_{\$} \{0,1\}$. Thus, $\mathcal{R}$ has a negligible advantage in winning the s-IND-MIFE-CPA game, meaning that MIFE is one-way secure. As a rapid consequence, there exists UBK-secure implementations for $\mathsf{PRP}_{K,n-m}(\cdot)$, for any randomly sampled K and for any choice of the fixed/free bits. □

On a separate note regarding the proof above, we remark that a very different theoretical approach would split the output of $\mathsf{PRP}_{K,n-m}(\cdot)$ as n concatenations of a puncturable PRF. Assuming that $\mathsf{pPRF} : \{0,1\}^{k+l} \times \{0,1\}^m \to \{0,1\}$, there exist (with overwhelming probability, a multiset of potentially larger) keys $K_1, \ldots, K_n$ to key pPRF such that:

$$\mathsf{PRP}_{K,n-m}(M) = \mathsf{pPRF}(K_1, M) || \ldots || \mathsf{pPRF}(K_n, M) \ .$$

By Lemma 5.6, such pPRFs admit UBK-secure implementations, and thus can be used in the proof of Lemma 5.7 instead of a lookup table.

Next, we show that assuming that UBK-secure implementation for $\mathsf{PRP}_{K,n-m}(\cdot)$ exist, then there exist a OW-MIFE-secure multi-input functional encryption scheme for $\mathsf{PRP}_{K,n-m-1}(\cdot)$, having $m+1 \leq n$ free variables. Applying this step $n-m$ times leads to the existence of UBK secure implementation for $\mathsf{PRP}_K(\cdot)$, via Lemma 5.2. However, the size of implementations obtained via BKS when instantiated with the scheme in Section 5.3.1.1 is not compact: the space complexity exceeds the one of a lookup table while "chaining" functions [GKP+13, p.23].

Lemma 5.8 (From $m \to m{+}1$ free inputs)**.** *Let* $\mathsf{PRP} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a secure pseudorandom permutation. Let* $\mathsf{C.Eval}_k^{r\,i}$ *denote an* UBK *secure implementation of* $\mathsf{PRP}_{K,i||n-m-1} : \{0,1\}^m \to \{0,1\}^n$*, where* $i \in \{0,1\}$ *and* $\mathsf{PRP}_{K,i||n-m-1}$ *denotes the restriction of the* PRP *keyed with a randomly sampled* K*, having* m *bits free, the* $m{+}1^{th}$ *bit set to* i *and* $n-m-1$ *bits fixed. Let* MIFE *be a full* s-IND-MIFE-CPA*-secure scheme with* $m{+}2$ *inputs defined over the message space* $\{0,1\} \times \ldots \times \{0,1\} \times \mathcal{K} \to \{0,1\}^n$ *supporting the circuit representation of* $\mathsf{PRP}_{K,n-m-1}$*. Then,* MIFE *is* OW-MIFE*-secure with respect to* $\mathsf{PRP}_{K,n-m-1}$ *and index set* $\{m+2\}$*.*

Proof. Again, we use a reduction to the full s-IND-MIFE-CPA game. As for the base case in Lemma 5.7, the reduction obtains the encryptions of 0 and 1 for each position $i \in [m+1]$. For position $m{+}2$, the reduction sends the real key K and a randomly sampled value.

The functional key query is executed with the following arguments: on the one hand, there is the real circuit computing $\mathsf{PRP}_{K,i||n-m-1}$. On the other hand, there is the circuit that contains the two UBK-secure implementations: $\mathsf{C.Eval}_k^{r\,0} || \mathsf{C.Eval}_k^{r\,1}$ and returns the output of

the relevant implementations. For the case of the BKS transform instantiated from [GKP+13], the sizes of the two implementations exceed the ones of a lookup-table, and we can safely set $\mathsf{C.Eval}_k^{r\,b}$ as LUT^b. This is due to the size of the msk_b^* for m bits of input, which is larger than the size of the lookup table with 2^m inputs. Equivalently, the advantage of an adversary in obtaining the keys from the lookup table implementations is negligible. The advantage of an adversary in recovering the secret key giving the BKS implementations obtained on top of the two implementations with m free bits is negligible.

Clearly, the two settings are equivalent, as they both describe the same $\mathsf{PRP}_{K,i||n-m-1}$. The reduction then emulates the OW-MIFE game with respect to an adversary $\mathcal{A}$, being able to answer its oracle queries. If the adversary extracts K with noticeable probability, $\mathcal{R}$ can distinguish between the two settings of the full s-IND-MIFE-CPA game. □

In the previous reduction, the size of the UBK implementation corresponding to case $m+1$ grows considerably, as it needs to support a functional-key for a circuit of size twice the size of the implementation corresponding to case m. To prevent a blow-up in the size parameters, we remark that the implementation corresponding to case $m+1$ supports a functional-key for the circuit representation of $\mathsf{PRP}_{K,n-m-1}$ padded with enough terms to match the sizes of the implementations of $\mathsf{PRP}_{K,0||n-m}$ and $\mathsf{PRP}_{K,1||n-m}$. As we would like a result to be generic, in this work we *assume* that the "compact" implementation supporting strictly the circuits in the class $\mathsf{PRP}_{K,n-m-1}$ is UBK-secure. Put differently, padding plays no role in the UBK-security and can be safely removed, obtaining a more compact implementation for $\mathsf{PRP}_{K,n-m-1}$.

One can also observe that BKS achieves a relaxed version of the s-IND-MIFE-CPA game, where a single challenge tuple of the form $\{(0,0),(1,1)\}$ is declared *a priori*, as well the key K of the PRP and the PRP circuit itself. Such a game would ask to distinguish between two settings: in the real setting, the functional key is used to compute the $\mathsf{PRP}_K(\cdot)$. In the ideal setting, the AGGs will use the UBK-secure implementations. More specifically, the BKS transform, when decrypting the ciphertexts in position one (corresponding to 0 and 1), returns two new functional keys, issued under msk_0^* and msk_1^*, which need to support $\mathsf{PRP}_{K,n-m-1}$. Let the BKS ciphertext corresponding to position 1 and input $M_1 = b \in \{0,1\}$ consist of the following *two* elements:

$$C_1^b \leftarrow_\$ \mathsf{FE}_1.\mathsf{Enc}(\mathsf{msk}_{out}, (\mathsf{msk}_b^*, K_b, 0)) \text{ and } \mathsf{sk}_1^b \leftarrow_\$ \mathsf{FE}_t.\mathsf{KGen}(\mathsf{msk}_{in}, AGG_{b,\perp,0,s,\mathsf{msk}_b^*,K_b})$$

An $m+2$-MIFE can be built on top of two $m+1$-MIFEs as follows: (1) take the above ciphertext corresponding to $M_1 = b$; (2) tweak the $AGG_{b,\perp,0,s,\mathsf{msk}_b^*||\mathsf{LUT},K_b}$ circuit to compute directly $\mathsf{PRP}_K(M_1||\ldots||M_m)$ via LUT and output an encryption of it

$$\Big(\mathsf{FE}_t.\mathsf{Enc}(\mathsf{msk}^*, (\mathsf{PRP}_K(M_1||\ldots||M_m), x_1, x_2), 1; r_1), \ldots, \mathsf{FE}_t.\mathsf{Enc}(\mathsf{msk}^*, x_{t+1}, t; r_t)\Big)$$

One can observe that the two settings of this weakened security experiment are functionally equivalent.

Finally, by repeating the previous argument, one can argue that if a PRP keyed with a randomly sampled K admits UBK secure implementations, there exists an MIFE construction that emulate the PRP with all its n-bit input being free (i.e. emulating $\mathsf{PRP}_K(\cdot)$).

Theorem 5.1. *Let* $\mathsf{PRP} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *be a secure pseudorandom permutation. Let* MIFE *be a full* s-IND-MIFE-CPA*-secure scheme with* $n+1$ *inputs defined over the message*

space $\{0,1\} \times \ldots \{0,1\} \times \mathcal{K} \rightarrow \{0,1\}^n$ *supporting the circuit representation of* PRP. *Then,* MIFE *is* OW-MIFE*-secure.*

Theorem 5.1. The proof follows as a consequence of the base case presented in Lemma 5.7 and of the transitory step from Lemma 5.8, as we can show that there exist two UBK-secure implementations for two PRPs having all but one bits of the input free: one having the fixed input set to 0 and the other set to 1. We can use the two implementations to feed in the MIFE scheme in contrast to having it running with respect to the real circuit that describes $\mathsf{PRP}(\cdot)$. Thus, down to the s-IND-MIFE-CPA of the scheme, the two settings are indistinguishable, meaning that an MIFE supporting $\mathsf{PRP}(K, \cdot)$ is OW-MIFE-secure. □

Informally, we remark that a different approach to tackle the space complexity issue may use an input slot to "encode" groups of r bits, instead of a single bit. In such a way, one can heavily decrease the arity of the MIFE scheme: from n to n/r supported inputs. But now, for each group of r bits there must be 2^r ciphertexts issued. We leave for future work the investigation of such a method built on top of a more efficient MIFE scheme [KS17].

5.3.2 Does iO-obfuscation of PRP Guarantee Their UBKSecurity?

Assume the existence of an iO obfuscator for a class of circuits taking l bits as input. Now, suppose there exists an UBK-secure implementation for a particular pseudorandom permutation (Section 5.3.1), the implementation not relying on the iO-security of the obfuscator. Let l stand for the size of this UBK-secure implementation.

5.3.2.1 Best Possible Obfuscator

As stated in [BGI+01], circuits for which there is no virtual black-box obfuscator, exist. The idea behind introducing best-possible obfuscation (BPO) [GR07; Chi15] was to capture what happens if *no* obfuscator can hide some specific information. Apparently, BPO would be an intermediate notion between VBB and iO. However, it turns out that it is, in fact, equivalent to iO [GR07; Chi15]. We provide below the definition of a BPO.

Definition 5.7 (Best Possible Obfuscator, [Chi15])**.** *A BPO obfuscator* $\mathcal{O}$ *is a* PPT *algorithm having the correctness and security defined identically to an indistinguishability obfuscator. Moreover, for any* PPT *adversary* $\mathcal{A}$*, there exists a* PPT $\mathcal{S}$ *such that: for all circuits* $\mathcal{C}_0, \mathcal{C}_1, \quad |\mathcal{C}_0| = |\mathcal{C}_1|$ *with* $\mathcal{C}_0 \equiv \mathcal{C}_1$ *we have that:*

$$\Pr[\mathcal{A}(\mathcal{O}(1^\lambda, \mathcal{C}_0)) \approx_c \mathcal{S}(1^\lambda, \mathcal{C}_1)]$$

belongs to $1 - \text{NEGL}(\lambda)$.

As we know from the work of Goldwasser *et al.* [GR07], iO is the *best possible obfuscator* (BPO), meaning that if some data is leaked on the circuit, then this leak is unavoidable. Stated equivalently, if the BPO does not hide some information, then that piece of information cannot be hidden by any equivalent representation of that program. Now, assume that applying $\mathsf{iO}(\mathsf{PRP}(K, \cdot)||\text{padding})$ does not hide the key K of the PRP (i.e. the obfuscated circuit is not UBK-secure). If this is the case, but at the same time we do have an equivalent implementation that is UBK-secure, then the iO obfuscator is not the best possible obfuscator for the class of PRP we consider, which contradicts the working hypothesis. Thus, it must

be the case an iO-obfuscation of $\mathsf{PRP}(K,\cdot)$ protects the key. We make use of this result – formalized in Corollary 5.1 – in the following part (Section 5.4), in the sense that the circuits that will be obfuscated will not leak the secret-keys embedded in them.

Corollary 5.1. *Let $\mathcal{E}$ be a secure pseudorandom-permutation and let $\mathsf{C}_{\mathcal{E}}.\mathsf{Eval}_r^k$ be a secure implementation w.r.t. $\mathcal{E}$ and a key K. Let iO be a sub-exponentially secure indistinguishability obfuscator supporting circuits of size $|\mathsf{C}_{\mathcal{E}}.\mathsf{Eval}_r^k|$ bits. Then, applying iO on the circuit describing $\mathcal{E}(K,\cdot)$ (appropriately padded) yields a UBK-secure implementation for $\mathcal{E}$ keyed by K.*

Corollary 5.1. It follows as a consequence of the work of Goldwasser and Rothblum [GR07] and by Theorem 5.1. Take the contrapositive and assume the program obtained via $\mathsf{iO}(\mathcal{C}_{\mathcal{E},K}(\cdot))$ does not hide the embedded key K, while we know about the existence of an equivalent program hiding it (Theorem 5.1). Then, there exists two PPT algorithms $\mathcal{A},\mathcal{S}$ such that a distinguisher can differentiate their outputs: $|\Pr[K \leftarrow\!\!\$\ \mathcal{A}(\mathsf{iO}(\mathcal{C}_{\mathcal{E},K}(\cdot)))] - \Pr[K \leftarrow\!\!\$\ \mathcal{S}(1^\lambda, \mathsf{C}.\mathsf{Eval}_K^r)]| \notin \text{NEGL}$. This contradicts the definition of BPO-security property. Finally, this means that iO is not the best possible obfuscator, which contradicts [GR07; Chi15]. □

5.4 UBK-Secure Implementations from iO and OWF

In this section, we employ iO to build UBK-secure implementations for pseudorandom permutations. Our proof relies on the *indistinguishability* property of the obfuscator. As we transit through a sequence of hybrids, we seek for formal guarantees that the obfuscated circuits do not leak information on the sensitive data they encapsulate, even when facing white-box adversaries.

5.4.1 A MIFE Scheme in the Public Setting

We commence with an overview of a multi-input functional encryption scheme. Proposed by Goyal, Jain and O'Neill [GJO16], it improves on the construction of Goldwasser *et al.* from [GGG+14]. The latter achieves public-key MIFE on top of iO. Finally, we prove that a slightly modified version of their construction provides an UBK-secure implementation for pseudorandom permutations. We point the reader to the original description for a complete, and perhaps more clear understanding of the scheme.

- $\mathsf{MIFE.Setup}$: assuming the arity of the supported functions is n, the Setup algorithm samples $2 \cdot n$, pairs of PKE keys:

$$(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda) \quad \text{and} \quad (\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)\ .$$

  Thus, each input $i \in [n]$ is associated with the two *pairs*. Moreover, a circuit $\mathcal{C}_i$ is built for each input:

$$\begin{aligned}
&\mathcal{C}_i \leftarrow \\
&\quad \textbf{Hardwire} : \mathsf{pk}_i^0, \mathsf{pk}_i^1, K_i \\
&\quad \textbf{Input} : c_i^0, c_i^1, M, R_i^0, R_i^1 \\
&\quad \textbf{Execute} : \\
&\quad \textbf{if } c_i^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_i^0, M; R_0) \vee c_i^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_i^1, M; R_1) \\
&\qquad \textbf{return } \perp \\
&\quad \textbf{return } \mathsf{PRF.Eval}(K_i, c_i^0 || c_i^1)
\end{aligned} \tag{5.1}$$

where the K_i corresponds to a PRF_i, and P corresponds to a one-way function. The circuit $\mathcal{C}_i$ is meant to provide a commitment to the two ciphertexts – through the evaluation of the PRF – under the hypothesis that the two ciphertexts have been correctly evaluated. $\mathcal{C}_i$ is then obfuscated as $\overline{\mathcal{C}_i} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_i)$.

Finally, the keys are set as:

$$\mathsf{msk} \leftarrow \bigcup_{i\in[n]} (\mathsf{sk}_i^0, \mathsf{sk}_i^1, K_i) \qquad \mathsf{mpk} \leftarrow \bigcup_{i\in[n]} (\mathsf{pk}_i^0, \mathsf{pk}_i^1, \overline{\mathcal{C}_i}, P)$$

- $\mathsf{MIFE.Enc}(\mathsf{mpk}, i, M)$: for position i, the MIFE ciphertext consists of two PKE-generated components encrypting the same plaintext M_i (for some $i \in [n]$), and a PRF evaluation over these ciphertexts:

$$c_i^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_i^0, M_i; R^0) \qquad c_i^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_i^1, M_i; R^1)$$

$$\pi_i \leftarrow\!\!\$\ \overline{\mathcal{C}_i}(c_i^0, c_i^1, M_i, R^0, R^1)$$

  Thus, the ciphertext is simply set as: $C_i \leftarrow (c_i^0, c_i^1, \pi_i)$ and this step is repeated for each of the n inputs of the MIFE scheme.

- A functional key sk_f for a function f is an obfuscation of a circuit that: (1) decrypts PKEs' ciphertext; (2) recovers the plaintext M; and (3) computes $f(M)$. In a sense, the secret-keys of the PKE are embedded in the obfuscated circuit.

$$\begin{aligned}
&\mathcal{C}_f \leftarrow \\
&\quad \textbf{Hardwire} : (\mathsf{sk}_i^0, K_i, P)_{i\in[n]} \\
&\quad \textbf{Input} : \{c_i^0, c_i^1, \pi_i\}_{i\in[n]} \\
&\quad \textbf{Execute} : \\
&\quad \textbf{for } i \leftarrow 1, n : \\
&\quad\quad \textbf{if } P(\mathsf{PRF.Eval}(K_i, c_i^0 || c_i^1)) \neq P(\pi_i) \\
&\quad\quad\quad \textbf{return } \perp \\
&\quad \textbf{return } f(\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0), \ldots, \mathsf{PKE.Dec}(\mathsf{sk}_n^0, c_n^0)) \\
&\mathsf{sk}_f \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_f)
\end{aligned} \tag{5.2}$$

- Decryption works in a straightforward manner, by feeding the obfuscated circuit of sk_f with the n inputs representing a ciphertext:

$$\mathsf{sk}_f((c_1^0, c_1^1, \pi_1), \ldots, (c_n^0, c_n^1, \pi_n)) \ .$$

5.4.2 UBK-Secure Implementations from [GJO16]

The spirit of the *unbreakability* security game is to prohibit the adversary from extracting an embedded key from a given implementation of a cryptographic primitive. We have seen that the notion of one-wayness we propose for multi-input functional encryption is shown to imply UBK. However, we observe the $\mathsf{OW\text{-}MIFE}$ game can be further relaxed in the public key setting, in the following sense: considering a $\mathsf{PRP} : \mathcal{K} \times M \to \mathcal{C}$, we can encrypt PRP's key K with respect to the first slot of the MIFE – say C_1 – and then encrypt the plaintext on the fly, under the second slot. Thus, only two slots are needed. We *stress* that, as C_1 is

generated once, there is no need to allow the adversary to query for a second $C_1' \neq C_1$. Put differently, in such a relaxed OW-MIFE game, the obfuscated circuit $\mathcal{C}$ occurring in [GJO16] accepts a single ciphertext for position 1.

From these reasons, we show directly how a UBK-secure implementation can be obtained from a simplified version of [GJO16], without transiting through the one-wayness of the original scheme.

$\underline{\mathsf{C.Setup}(1^\lambda, K, \mathcal{E})}$:
for $i \leftarrow 1$ to 2:
$K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
$(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0||c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$
$\mathcal{C}_2 \leftarrow \ldots$ (Right Column)
$\mathsf{sk}_\mathcal{E} \leftarrow \ldots$ (Right Column)
$\mathsf{C.Eval}_K^r \leftarrow (\mathsf{sk}_\mathcal{E}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
return $\mathsf{C.Eval}_K^r$

$\underline{\mathsf{C.Eval}_K^r(M)}$:
Hardwire : $(\mathsf{sk}_\mathcal{E}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$c_2^0 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R^0)$
$c_2^1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R^1)$
$\pi_2 \leftarrow \overline{\mathcal{C}_2}(c_2^0, c_2^1, M, R^0, R^1)$
return $\mathsf{sk}_\mathcal{E}((c_2^0, c_2^1, \pi_2))$

$\mathcal{C}_2 \leftarrow$
Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
Execute:
if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
$\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
return $\perp$
return $(\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_\mathcal{E} \leftarrow$
Hardwire: $(\mathsf{sk}_i^0, K_i)_{i\in[2]}, C_1$
Input: $\{c_2^0, c_2^1, \pi_2\}$
Execute:
if $\mathsf{PRF.Eval}(K_i, c_i^0||c_i^1) \neq \pi_i$
return $\perp$
return $f(\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0), \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_\mathcal{E})$

Figure 5.12: A slight modification of the scheme in [GJO16]. A generic construction of a UBK-compiler from sub-exponentially secure iO, public-key encryption and pseudorandom function. In [GJO16, Theorem 3], the authors prove the construction achieves indistinguishability. The PRF is defined over $\mathcal{K}_\mathsf{PRF} \times (\mathcal{C} \times \mathcal{C}) \to \mathcal{Y}$. As we only seek for one-wayness, we do not require the puncturability from the PRF. The PKE schemes are defined over $\mathcal{K}_\mathsf{PKE} \times \mathcal{M} \to \mathcal{C}$, while $\mathcal{O}$ is a sub-exponentially-secure indistinguishability obfuscator. The correctness of the construction follows similarly to the one of the MIFE in [GJO16].

Our proof for one-wayness covers the case of two input functions solely. This suffices for building white-box implementations of PRPs (see Lemma 5.1) and allows a relatively clear exposition. However, it is not hard to derive a proof for the general case of n-input MIFE schemes in a similar manner.

Theorem 5.2 (UBK implementation from Figure 5.12)**.** *Let $\mathcal{O}$ denote a circuit obfuscator enjoying indistinguishability and let $\mathcal{E}$ be the circuit representation of $f : \mathcal{K} \times M \to \mathcal{C}$, a secure pseudorandom permutation. Let $\mathsf{C.Eval}_k^r$ denote the implementation corresponding to $\mathcal{E}$ described in Figure 5.12. For any* PPT *adversary $\mathcal{A}$, the advantage in winning the unbreakability security experiment (Definition 5.2) with respect to $\mathcal{E}$ is bounded as follows:*

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{R}_1,\mathsf{PKE}}(\lambda) + 3 \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathcal{R}_2,\mathcal{O}}(\lambda) + \mathsf{Adv}^{\mathsf{owf}}_{\mathcal{R}_3,\mathcal{E}}(\lambda) \ .$$

Theorem 5.2. First, notice the particularity in the construction introduced by Goyal *et al.* , where two sets of public-keys are used to encrypt the input, which in a sense helps to conceive a solution. The proof follows from a hybrid argument, in rough terms its overview is summarized as follows: first, we switch from encrypting K to K^* with respect to pk_1^1. Then, we change the form of the KGen procedure, ending up with a circuit where the inner decryption step runs with respect to K rather than to (c_1^0, sk_0). Once in this state, we switch from encrypting K to K^* with respect to pk_0. In the last game, we bound the advantage of an adversary in winning the one-wayness game by the advantage of winning the one-wayness game against the underlying PRP, i.e. $\mathcal{E}$.

Hybrid Games' Description. A game-based description is given in Figures 5.13 and 5.14, allowing to follow the changes that are made between two games easily.

- Game_0: the first game corresponds to the real UBK security experiment cf. Definition 5.2.
- Game_1: is identical to Game_0, up to the encryption of the real message under pk_1^1. We change from encrypting K to a randomly sampled K^*:

  $$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*) \ .$$

  The distance to the previous game is bounded by $\mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{A},\mathsf{PKE}}(\lambda)$ from Game_0.
- Game_2: we rely on the indistinguishability of the obfuscator in order to wire-in the secret K in circuit $\mathcal{C}_\mathcal{E}$. The decryption step in $\mathcal{C}_\mathcal{E}$ is then changed by using K directly rather than executing $\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0)$. We note this circuit is equivalent to the one in Game_1, as its behaviour is preserved when adding the extra constant K to compute the output of $\mathcal{E}$.
- Game_3: is identical to the previous one, except that we change the derivation procedure. The change consists in removing sk_i^0 from the set of hardwired constants; thus the two circuits being functionally equivalent. We note the iO-security of the obfuscator bounds this game hop.
- Game_4: is identical to the previous game, except that we remove (c_1^0, c_1^1, π_1) from the description of $\mathcal{C}_\mathcal{E}$. Note that this step is permitted, as the aforementioned values are no longer used in $\mathcal{C}_\mathcal{E}$ while these two descriptions of $\mathcal{C}_\mathcal{E}$ remain equivalent.
- Game_5: identical to the previous game, except the fact that we change c_1^0 to

  $$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K^*) \ .$$

  Finally, in this setting, we are able to bound the advantage of an adversary in winning the UBK game, by the advantage of an adversary which wins the *one-wayness* game against the underlying pseudorandom permutation $\mathcal{E}$.

Lemma 5.9 ($\text{Game}_0 \to \text{Game}_1$)**.** *For any* PPT *distinguisher* $\mathcal{D}$*,*

$$|\Pr[\text{Game}_0^\mathcal{D}] - \Pr[\text{Game}_1^\mathcal{D}]| \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{R},\mathsf{PKE}}(\lambda) \ .$$

$\underline{\text{Game}_0 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_\mathcal{E}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0||c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_\mathcal{E} \leftarrow$
  Hardwire: $(\mathsf{sk}_i^0, K_i)_{i\in[2]}, (c_1^0, c_1^1, \pi_1)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1) \neq \pi_2$
    return $\perp$
  return $f(\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0), \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_\mathcal{E})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_\mathcal{E}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_\mathcal{E}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

$\underline{\text{Game}_1 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_\mathcal{E}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0||c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_\mathcal{E} \leftarrow$
  Hardwire: $(\mathsf{sk}_i^0, K_i)_{i\in[2]}, (c_1^0, c_1^1, \pi_1)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1) \neq \pi_2$
    return $\perp$
  return $f(\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0), \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_\mathcal{E})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_\mathcal{E}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_\mathcal{E}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

$\underline{\text{Game}_2 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_\mathcal{E}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0||c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_\mathcal{E} \leftarrow$
  Hardwire: $(\mathsf{sk}_2^0, K_i)_{i\in[2]}, K, (c_1^0, c_1^1, \pi_1)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0||c_2^1) \neq \pi_2$
    return $\perp$
  return $f(K, \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_\mathcal{E})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_\mathcal{E}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_\mathcal{E}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

Figure 5.13: The hybrid experiments $\text{Game}_0 \rightarrow \text{Game}_2$.

$\underline{\text{Game}_3 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_{\mathcal{E}}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0 || c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_{\mathcal{E}} \leftarrow$
  Hardwire: $(\mathsf{sk}_2^0, K_1, K_2, K, c_1^0, c_1^1, \pi_1)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1) \neq \pi_2$
    return $\perp$
  return $f(K, \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_{\mathcal{E}} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_{\mathcal{E}})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_{\mathcal{E}}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_{\mathcal{E}}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

$\underline{\text{Game}_4 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_{\mathcal{E}}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0 || c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_{\mathcal{E}} \leftarrow$
  Hardwire: $(\mathsf{sk}_2^0, K_1, K_2, K)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1) \neq \pi_2$
    return $\perp$
  return $f(K, \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_{\mathcal{E}} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_{\mathcal{E}})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_{\mathcal{E}}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_{\mathcal{E}}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

$\underline{\text{Game}_5 \quad \mathsf{UBK}^{\mathcal{A}}_{\mathsf{C}_{\mathcal{E}}}(\lambda)}$:

$(R^0, R^1) \leftarrow\!\!\$\ \mathcal{R}$
$K \leftarrow\!\!\$\ \mathcal{K}$

for $i \leftarrow 1$ to 2:
  $K_i \leftarrow\!\!\$\ \mathsf{PRF.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^0, \mathsf{sk}_i^0) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
  $(\mathsf{pk}_i^1, \mathsf{sk}_i^1) \leftarrow\!\!\$\ \mathsf{PKE.Setup}(1^\lambda)$
$c_1^0 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^0, K^*; R^0)$
$c_1^1 \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}_1^1, K^*; R^1)$
$\pi_1 \leftarrow\!\!\$\ \mathsf{PRF.Eval}(K_1, c_1^0 || c_1^1)$
$C_1 \leftarrow (c_1^0, c_1^1, \pi_i)$

$\mathcal{C}_2 \leftarrow$
  Hardwire: $\mathsf{pk}_2^0, \mathsf{pk}_2^1, K_2$
  Input: $c_2^0, c_2^1, M, R_2^0, R_2^1$
  Execute:
    if $c_2^0 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^0, M; R_0) \vee$
        $\vee c_2^1 \neq \mathsf{PKE.Enc}(\mathsf{pk}_2^1, M; R_1)$
      return $\perp$
    return $(\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1))$
$\overline{\mathcal{C}_2} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_2)$

$\mathcal{C}_{\mathcal{E}} \leftarrow$
  Hardwire: $(\mathsf{sk}_2^0, K_1, K_2, K)$
  Input: $\{c_2^0, c_2^1, \pi_2\}$
  Execute:
  if $\mathsf{PRF.Eval}(K_2, c_2^0 || c_2^1) \neq \pi_2$
    return $\perp$
  return $f(K, \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$
$\mathsf{sk}_{\mathcal{E}} \leftarrow\!\!\$\ \mathcal{O}(\mathcal{C}_{\mathcal{E}})$

$\mathsf{C.Eval}_k^r \leftarrow (\mathsf{sk}_{\mathcal{E}}, \overline{\mathcal{C}_2}, \mathsf{pk}_2^0, \mathsf{pk}_2^1)$
$K' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{C}_{\mathcal{E}}}(1^\lambda, \mathsf{C.Eval}_k^r)$
return $K = K'$

Figure 5.14: The hybrid experiments $\text{Game}_3 \rightarrow \text{Game}_5$.

Lemma 5.9. Overall, we begin by considering the format of a ciphertext C_1 in Figure 5.12, which can be seen as a triplet consisting of two PKE ciphertexts (c_1^0, c_1^1) obtained under two different public-keys, and a PRF value, essentially acting as a commitment to (c_1^0, c_1^1). The game replaces the values c_1^1 and the corresponding PRF evaluation with encryptions of K^* relying on IND-CPA of PKE. The reduction works as follows: assume the IND-CPA security experiment where $(\mathsf{pk}_1^1, \mathsf{sk}_1^1)$ are sampled. Assume the existence of a PPT adversary $\mathcal{A}$ that can distinguish between Game_0 and Game_1. We build a PPT algorithm $\mathcal{R}$, that wins the IND-CPA game as follows:

- The IND-CPA game samples $(\mathsf{pk}_1^1, \mathsf{sk}_1^1)$ and provides pk_1^1 to $\mathcal{R}$. $\mathcal{R}$ uses the given pk_1^1 and has no knowledge on sk_1^1.
- $\mathcal{R}$ samples $(K, K^*) \leftarrow\!\!\$\ \mathcal{K} \times \mathcal{K}$ as the challenge messages.
- $\mathcal{R}$ provides the challenge messages to the IND-CPA experiment and obtains the challenge ciphertext c_1^1.
- $\mathcal{R}$ includes the challenge c_1^1 as part of the input for the adversary $\mathcal{A}$. Note that c_1^0 is perfectly computable (since pk_1^0 is known). Similarly, π_1 is computable. Also, note that $\mathcal{C}_\mathcal{E}$ can be computed since all sk_1^0 are known; thus $\mathcal{R}$ is able to derive $\mathsf{sk}_\mathcal{E}$.

If with a certain advantage ϵ, a distinguisher distinguishes between the two game settings and returns a bit $b_\mathcal{A}$, then $\mathcal{R}$ returns the same value $b_\mathcal{A}$ as its output. □

Lemma 5.10 ($\text{Game}_1 \to \text{Game}_2$)**.** *For any* PPT *distinguisher* $\mathcal{D}$*,*

$$|\Pr[\text{Game}_1^\mathcal{D}] - \Pr[\text{Game}_2^\mathcal{D}]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathcal{R},\mathcal{O}}(\lambda) \ .$$

Lemma 5.10. Game_2 is indistinguishable from Game_1 down to the indistinguishability property of the obfuscator $\mathcal{O}$. Note that in Game_2, we wire-in K in $\mathcal{C}_\mathcal{E}$.

The reduction $\mathcal{R}$ builds two functionally equivalent circuits — one corresponding to the setting in Game_1, the other hardwiring extra values such as K and returning $\mathcal{E}(K, M)$ if M is queried — and get from the iO game an obfuscation of one of the two circuits. In detail, $\mathcal{R}$ samples the corresponding $\mathsf{PKE}/\mathsf{PRF}$ keys for all indexes, then changes $\mathcal{C}_\mathcal{E}$ to compute $f(K, \cdot)$. It is clear that $\mathcal{R}$ can simulate both games forwarding the two functionally equivalent circuits to the obfuscator and getting back an obfuscated circuit. Also, we stress the two circuits are functionally equivalent, as (1) the sanity check:

$$\mathsf{PRF.Eval}(K_i, c_i^0, c_i^1) \neq \pi_i$$

is the same in both circuits; (2) when the adversary queries C_2, both circuits return:

$$f(K, \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0)) = f(\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0), \mathsf{PKE.Dec}(\mathsf{sk}_2^0, c_2^0))$$

as we know that $\mathcal{R}$ crafts the challenge ciphertext C^* to fulfil the constraint:

$$\mathsf{PKE.Dec}(\mathsf{sk}_1^0, c_1^0) = K \ .$$

Thus, assuming the existence of a distinguisher $\mathcal{D}$ between Game_1 and Game_2, $\mathcal{R}$ can use $\mathcal{D}$ to win the iO game. To this end, $\mathcal{R}$ returns whatever bit b is obtained from $\mathcal{D}$. Clearly, if $\mathcal{D}$ can distinguish between the hybrids, $\mathcal{R}$ wins the iO game with a similar advantage. □

Lemma 5.11 ($\text{Game}_2 \to \text{Game}_3$)**.** *For any* PPT *distinguisher* $\mathcal{D}$*,*

$$|\Pr[\text{Game}_2^{\mathcal{D}}] - \Pr[\text{Game}_3^{\mathcal{D}}]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathcal{R},\mathcal{O}}(\lambda) \ .$$

Lemma 5.11. The transition from Game_2 to Game_3 is based, again, on the iO property of the obfuscator $\mathcal{O}$. First, we argue that the circuits corresponding to $\mathcal{C}_{\mathcal{E}}$ in Game_2 and Game_3 are functionally equivalent; this is straightforward as the only change consists in removing the constant sk_1^0, which is no longer used in the circuit.

As regards to the reduction, it proceeds as for the previous hop. First, $\mathcal{R}$ samples all the required data to simulate the UBK game. From then on, the reduction is similar to the previous case. Considering that the iO game samples uniformly at random a circuit out of the two classes and this $\mathcal{C}_f$ is given to $\mathcal{R}$, then $\mathcal{R}$ simulates either Game_2 or Game_3. Depending on the bit b issued by a distinguisher $\mathcal{D}$, $\mathcal{R}$ wins the indistinguishability game. □

Lemma 5.12 ($\text{Game}_3 \to \text{Game}_4$)**.** *For any* PPT *distinguisher* $\mathcal{D}$*,*

$$|\Pr[\text{Game}_3^{\mathcal{D}}] - \Pr[\text{Game}_4^{\mathcal{D}}]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathcal{R},\mathcal{O}}(\lambda) \ .$$

Lemma 5.12. The transition from Game_3 to Game_4 is based, again, on the iO property of the obfuscator $\mathcal{O}$. Concretely, the circuits corresponding to $\mathcal{C}_{\mathcal{E}}$ in Game_3 and Game_4 are equivalent up to the usage of the ciphertext (c_1^0, c_1^1, π_1). As it can be easily observed, (c_1^0, c_1^1, π_1) plays no role in the description of $\mathcal{C}_{\mathcal{E}}$ and can be safely detached from the set of hard-wired values. A reduction $\mathcal{R}$ may use a distinguisher $\mathcal{D}$ noticing this change in breaking the iO game. □

Lemma 5.13 ($\text{Game}_4 \to \text{Game}_5$)**.** *For any* PPT *distinguisher* $\mathcal{D}$*,*

$$|\Pr[\text{Game}_4^{\mathcal{D}}] - \Pr[\text{Game}_5^{\mathcal{D}}]| \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{R},\mathsf{PKE}}(\lambda) \ .$$

Lemma 5.13. The transition is based virtually on the same argument used to motivate Lemma 5.9, thus we defer it here. □

Lemma 5.14 (Advantage in Game_5)**.** *For any* PPT *adversary* $\mathcal{A}$*,*

$$\Pr[\text{Game}_5^{\mathcal{A}}] \leq \mathsf{Adv}^{\mathsf{owf}}_{\mathcal{R},f}(\lambda) \ .$$

Lemma 5.14. Finally, in Game_5, the challenge ciphertext corresponds to the encryption of K^*, while the winning condition asks the adversary to return K, having access to the values of $f(K, \cdot)$. We note the value of K is hardwired in the obfuscated circuit and an adversary cannot extract it. As shown in Corollary 5.1 (Section 5.3), since a UBK-implementation not using iO already exists, and based on the fact that iO is the best possible obfuscator, the implementation will not leak K. We also note that based on this property, sk_i, K_i are not leaked (otherwise it could have directly extracted sks and decrypt the PKE ciphertext). The setting maps perfectly to a OWF-game corresponding to $f(K, \cdot)$, where the adversary can query f for at various points. Thus, if an adversary $\mathcal{R}$ recovers K, it breaks the one-wayness of the scheme. □

Thus, the adversary wins if it guesses at least a component. By applying the union bound we conclude that:

$$\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{MIFE}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\text{ind-cpa}}_{\mathcal{R}_1,\mathsf{PKE}}(\lambda) + 3 \cdot \mathsf{Adv}^{\mathsf{io}}_{\mathcal{R}_2,\mathcal{O}}(\lambda) + \mathsf{Adv}^{\mathsf{owf}}_{\mathcal{R}_3,f}(\lambda) \ .$$

This completes the proof of Theorem 5.2. □

Chapter 6

Functional Encryption with Auxiliary Inputs and WBC

In this chapter, we put forward the notion of *functional encryption with auxiliary inputs*, abbreviated FEAI. It can be thought of as a generalization of the classical FE primitive with the key difference that the decryption algorithm takes an unencrypted value as an auxiliary input. Our definitions are for both the public and private key settings. As the main security notion, we focus on indistinguishability while we show that achieving it suffices to obtain an *indistinguishability obfuscator*. Furthermore, we show that one-way secure FEAI is instrumental in achieving UBK security.

6.1 Functional Encryption with Auxiliary Input

Syntactically, a *functional encryption with auxiliary inputs* (FEAI) scheme supports two-input functions. Its distinction from a two-input functional encryption resides in the *second input* – say aux– which is provided for the decryption procedure in an *unencrypted* format. Given a ciphertext encrypting M and a functional key for f, the decryptor then recovers $f(M, \mathsf{aux})$ for any aux in the input domain of f. Despite its simplicity and the fact that it may appear as folklore, to the best of our knowledge, this is the first time that such a primitive is formalized.

Indistinguishability. Naturally, in order to avoid attacks coming from a trivial distinguisher, the classical indistinguishability security notion must enforce that $f(M_1, \mathsf{aux}) = f(M_2, \mathsf{aux})$ for *any* possible combination of auxiliary inputs an adversary may choose. Although such a constraint may seem to restrict the number of supported functionalities, we point out that certain primitives – such as puncturable PRFs [SW14] – may be FEAI-suitable.

One-Wayness. A second security notion that we propose, namely one-wayness, captures the ability of the scheme to hide the encrypted message in the presence of a functional key, issued for a one-way (candidate) function. Put differently, it provides an adversary with a ciphertext of a randomly sampled message M, while we require that finding M to be hard in the presence of a functional key sk_f as long as f is one-way.

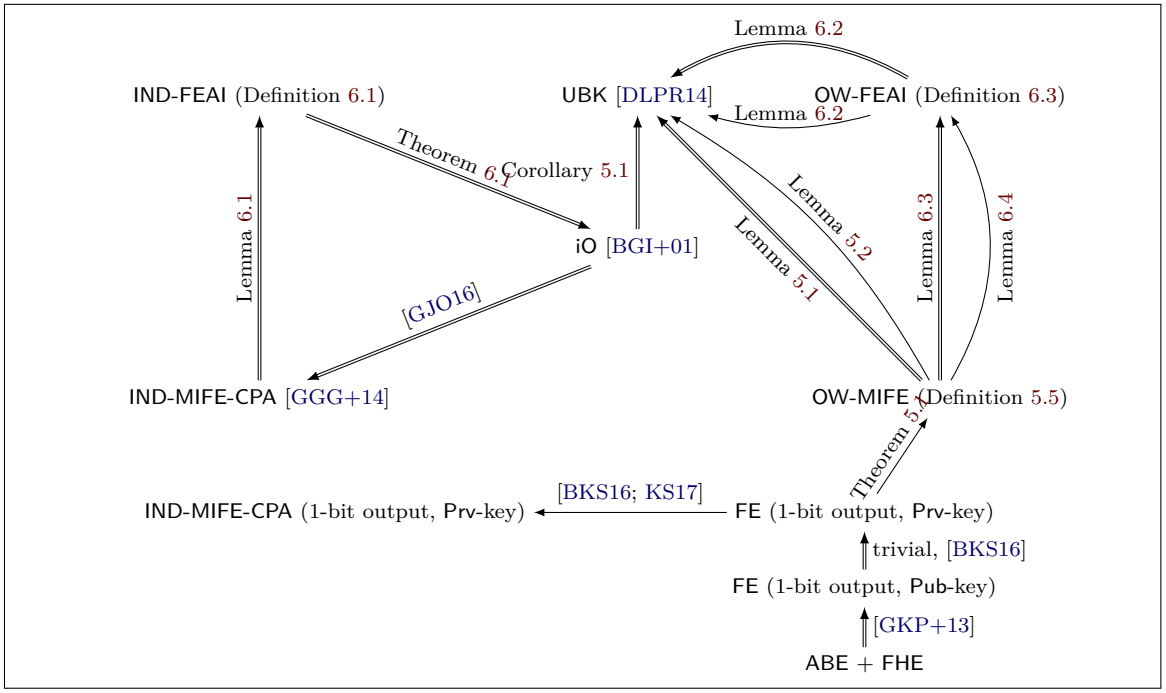

Figure 6.1: A graphical depiction of the relations between the security notions we consider in achieving UBK-security. FEAI is characterized through indistinguishability and one-wayness.

Regarding the structure of this chapter, we begin by introducing the definitions for FEAI in Section 6.2, and then continue by relating them to UBK-security (Section 6.3) and OW-MIFE (Section 6.4).

6.2 Definitions

6.2.1 Public-Key Setting

The definition of FEAI in the public case follows. Regarding the indistinguishability security notion we present, it allows the adversary to obtain functional keys under the restriction that $f(M_0, \mathsf{aux}) = f(M_1, \mathsf{aux})$ for any message aux in the domain of f where (M_0, M_1) represent the challenge messages the adversary submits during the security experiment.

Definition 6.1 (Functional Encryption with Auxiliary Input - Public-Key Setting)**.** *Let $\mathcal{F}_\lambda = \{f \mid f : \mathcal{M}_\lambda \times \mathcal{X}_\lambda \to \mathcal{V}_\lambda\}$ be an ensemble of two-input functions. A functional encryption with auxiliary input scheme consists of four algorithms* (Setup, KGen, Enc, Dec) *such that:*

- *The* Setup *algorithm* $(\mathsf{msk}, \mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *takes the security parameter λ and outputs a master secret key* msk *and the master public key* mpk*;*
- *The encryption algorithm* $C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, M)$ *takes as input the encryption key* mpk *and a message $M \in \mathcal{M}$, and outputs a ciphertext C;*
- *The functional key derivation algorithm* $\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f)$ *takes as input the description of a function $f \in \mathcal{F}_\lambda$ and outputs the corresponding functional key* sk_f*;*
- *The decryption algorithm* $\{f(M, \mathsf{aux}), \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}_f, C, \mathsf{aux})$ *is a deterministic algorithm that takes as input a functional key* sk_f*, a ciphertext C and an auxiliary input* aux*, and outputs a string $f(M, \mathsf{aux})$ or a special error symbol $\perp$.*

We define the following properties of an FEAI *scheme:*

- ***Correctness**: for all $\mathsf{aux} \in \mathcal{X}$ and for all $M \in \mathcal{M}$ the following holds:*

$$\Pr\left[\begin{array}{r|l} \mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{mpk}, M), \mathsf{aux}) & (\mathsf{msk}, \mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \wedge \\ = f(M, \mathsf{aux}) & \mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f) \end{array}\right] \in \text{1-}\textsc{Negl}(\lambda)\ .$$

- ***Indistinguishability**: for any* PPT *adversary $\mathcal{A}$, the following advantage*

$$\mathsf{Adv}^{\mathsf{ind\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) := \left|\Pr[\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda) = 1] - \frac{1}{2}\right|$$

is negligible, where the security experiment $\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda)$ *is defined in Figure 6.2.*

$\underline{\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{FEAI.Setup}(1^\lambda)$
$(M_0, M_1) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}_{\mathsf{msk}}(\cdot)}(1^\lambda, \mathsf{mpk})$
$C^* \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, M_b)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}_{\mathsf{msk}}(\cdot)}(1^\lambda, \mathsf{mpk}, C^*)$
return $b = b' \wedge \mathrm{Valid}(\mathrm{L}, M_0, M_1)$

$\underline{\mathsf{KGen}_{\mathsf{msk}}(f)}$:
$\mathrm{L} \leftarrow \mathrm{L} \cup \{f\}$
return $\mathsf{FEAI.KGen}(\mathsf{msk}, f)$

$\underline{\mathrm{Valid}(\mathrm{L}, M_0, M_1)}$:
if $\exists \mathsf{aux}, \exists f \in L$ s.t. $f(M_0, \mathsf{aux}) \neq f(M_1, \mathsf{aux})$
  return 0
return 1

Figure 6.2: The (adaptive) IND-FEAI security experiment in the public-key setting.

The definition for the private-key setting is similar, in some sense keeping a parallel to the MIFE definitions. We postpone it to Section 6.2.2 while focusing on the relations involving public-key FEAI.

6.2.1.1 IND-FEAI Implies Indistinguishability Obfuscation.

In this part, we look into the relation between FEAI and iO. First, we show that FEAI achieving indistinguishability implies the existence of a trivial construction of an indistinguishable obfuscator.

$\underline{\mathsf{iO}(\mathcal{C}):}$
$\widetilde{\mathcal{C}} \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, \mathcal{C})$
$P := \mathsf{dec}(\mathsf{sk}_{U_\lambda}, \widetilde{\mathcal{C}}, \cdot)$
return P

Figure 6.3: An indistinguishability obfuscator on top of an FEAI scheme. Evaluating the obfuscated program iO on input x is done as follows: $\mathsf{iO}_P(x) := \mathsf{FEAI.dec}(\mathsf{sk}_U, \mathcal{C}, x)$. By correctness, the result is $\mathcal{C}(x)$.

Theorem 6.1. *Assuming the existence of an* IND-FEAI *secure functional encryption with auxiliary inputs scheme* $\mathsf{FEAI} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{dec})$ *supporting functions represented as circuits* $\mathcal{C}_\lambda$*, the construction in Figure 6.3 is an indistinguishability obfuscator* iO *for* $\mathcal{C}_\lambda$ *such that* $\mathsf{Adv}^{\mathsf{io}}_{\mathcal{A},\mathsf{iO}}(\lambda) \leq \mathsf{Adv}^{\text{ind-feai}}_{\mathcal{R},\mathsf{FEAI}}(\lambda)$.

Theorem 6.1. Let $\mathcal{C}_\lambda$ be the class of circuits of size at most $poly(\lambda)$ for some polynomial $poly$. Let U_λ denote a universal circuit for $\mathcal{C}_\lambda$, where $U_\lambda(C, M) = C(M)$ for any $C \in \mathcal{C}_\lambda$ and any input $M \in \mathcal{M}$. Given an FEAI instance $(\mathsf{msk}, \mathsf{mpk})$ sampled via $\mathsf{Setup}(1^\lambda)$, a functional key is derived for the universal circuit U_λ by $\mathsf{sk}_{U_\lambda} \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, U_\lambda)$. The reduction R constructs the iO scheme described in Figure 6.3.

We denote by $\mathcal{A}$ a distinguisher having a non-negligible advantage in winning the indistinguishability game for iO. $\mathcal{A}$ sends two functionally equivalent circuits $\mathcal{C}_0, \mathcal{C}_1$ to the reduction $\mathcal{R}$. The reduction sends the two circuits as messages to the IND-FEAI game and obtains $\widetilde{C}$ from the IND-FEAI game, as well as a functional key for U_λ such that $U_\lambda(\mathcal{C}_0, \cdot) = U_\lambda(\mathcal{C}_1, \cdot)$. Then $\mathcal{R}$ provides $\mathcal{A}$ with $(\mathsf{sk}_{U_\lambda}, \widetilde{C})$ and wins the IND-FEAI game under the same advantage $\mathcal{A}$ does. □

IND-FEAI from IND-MIFE-CPA. We note that a MIFE (a two-input scheme suffices), can be readily transformed into a FEAI scheme. The encryption key for the second input should be public, while the first encryption key could be private or public depending on the applications. The indistinguishability of our construction follows immediately from the one of MIFE.

$\underline{\mathsf{FEAI.Setup}(1^\lambda):}$
$(\mathsf{msk}, \mathsf{mpk}_1, \mathsf{mpk}_2) \leftarrow\!\!\$\ \mathsf{2in\text{-}FE.Setup}(1^\lambda)$
$\mathsf{mpk} \leftarrow (\mathsf{mpk}_1, \mathsf{mpk}_2)$
return $(\mathsf{msk}, \mathsf{mpk})$

$\underline{\mathsf{FEAI.KGen}(\mathsf{msk}, f):}$
return $\mathsf{2in\text{-}FE.KGen}(\mathsf{msk}, f)$

$\underline{\mathsf{FEAI.Enc}(\mathsf{mpk}, M):}$
return $\mathsf{2in\text{-}FE.Enc}(\mathsf{mpk}_1, M)$

$\underline{\mathsf{FEAI.Dec}(\mathsf{sk}_f, C, \mathsf{aux}):}$
$C_{\mathsf{aux}} \leftarrow\!\!\$\ \mathsf{2in\text{-}FE.Enc}(\mathsf{mpk}_2, \mathsf{aux})$
return $\mathsf{2in\text{-}FE.Dec}(\mathsf{sk}_f, C, C_{\mathsf{aux}})$

Figure 6.4: The natural construction of FEAI from 2in-FE where mpk_2 is public.

Lemma 6.1 (IND-MIFE-CPA ⇒ IND-FEAI)**.** *Let 2in-FE be a two-input functional encryption scheme that enjoys* IND-MIFE-CPA *security against any adversary $\mathcal{R}$. Then, the FEAI construction in Figure 6.4 achieves indistinguishability against any* PPT *adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}mife\text{-}cpa}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 6.1. The proof is immediate. Let $\mathcal{A}$ be a PPT adversary able to break the IND-FEAI game against FEAI. We show how to build an adversary $\mathcal{R}$ against the IND-MIFE-CPA security of the underlying 2in-FE scheme. First, the IND-MIFE-CPA game samples the pair of keys:

$$(\mathsf{msk}, \mathsf{mpk}_1, \mathsf{mpk}_2) \leftarrow\!\!\$\ \text{2in-FE.Setup}(1^\lambda),$$

and gives $(\mathsf{mpk}_1, \mathsf{mpk}_2)$ to $\mathcal{R}$ (which forwards the second to $\mathcal{A}$, as the set up for FEAI scheme). $\mathcal{A}$ selects a pair of challenge messages (M_0, M_1) and sends them to $\mathcal{R}$, who regards them as the challenge messages corresponding to the first input, then $\mathcal{R}$ sends all the challenge messages to the IND-MIFE-CPA game. The challenger returns the ciphertext C_1^b to $\mathcal{R}$, who forwards to $\mathcal{A}$ the pair (C_1^b, mpk_2). Any functional key request made by $\mathcal{A}$ is forwarded by $\mathcal{R}$ to the IND-MIFE-CPA challenger[1]. Finally, $\mathcal{A}$ replies with b', which is forwarded by $\mathcal{R}$ to the IND-MIFE-CPA challenger. It is easy to see that if $\mathcal{A}$ guesses b, then $\mathcal{R}$ returns a correct answer under the same advantage. Thus, $\mathsf{Adv}^{\mathsf{ind\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}mife\text{-}cpa}}_{\mathcal{R},\mathsf{MIFE}}(\lambda)$. □

6.2.2 FEAI - Private-Key Setting.

FEAI in the private-key setting is similar to the MIFE counterpart. A master encryption key is derived, to be used during the key-derivation and ciphertext-generation procedures. Subsequent changes need to be enforced in the indistinguishability experiment: in order to prevent the adversary trivially winning the game, we need to enforce that for all functions f for which functional-keys have been issued, it must be the case that "$f(M_0, \cdot) = f(M_1, \cdot)$". If this is not the case, it becomes trivial for an adversary to win this game, given that it can craft M_0 and M_1 such that for a specific input X, $f(M_0, X)$ has a certain value.

Definition 6.2 (Functional Encryption with Auxiliary Input - Private-Key Setting)**.** *Let $\mathcal{F}_\lambda = \{f \mid f : \mathcal{M}_\lambda \times \mathcal{X}_\lambda \to \mathcal{V}_\lambda\}$ be an ensemble of two-input functions. A functional encryption with auxiliary input consists of four algorithms* (Setup, KGen, Enc, Dec) *such that:*

- *The* Setup *algorithm* $\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$ *takes the security parameter λ and outputs a master secret key* msk*;*
- *The encryption algorithm* $C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, M)$ *takes as input the encryption key* msk *and a message $M \in \mathcal{M}$, and outputs a ciphertext C;*
- *The functional key derivation algorithm* $\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f)$ *takes as input the description of a function $f \in \mathcal{F}_\lambda$ and outputs the corresponding functional key* sk_f*;*
- *The decryption algorithm* $\{f(M, \mathsf{aux}), \perp\} \leftarrow \mathsf{Dec}(\mathsf{sk}_f, C, \mathsf{aux})$ *is a deterministic algorithm that takes as input a functional key* sk_f*, a ciphertext C and an auxiliary input* aux*, and outputs a string* $f(M, \mathsf{aux})$ *or a special error symbol $\perp$.*

[1]Note that a valid sk_f request made by $\mathcal{A}$ implies a valid sk_f request made by $\mathcal{R}$.

$\underline{\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda)}$:
$\mathrm{L} \leftarrow \emptyset$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(M_0, M_1) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}_{\mathsf{msk}}(\cdot),\mathsf{Enc}_{\mathsf{msk}}(\cdot)}(1^\lambda)$
$C^* \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, M^b)$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{KGen}_{\mathsf{msk}}(\cdot),\mathsf{Enc}_{\mathsf{msk}}(\cdot)}(1^\lambda, C^*)$
return $b = b' \wedge \mathrm{Valid}(\mathrm{L}, M_0, M_1)$

$\underline{\mathsf{KGen}_{\mathsf{msk}}(f)}$:
$\mathrm{L} \leftarrow \mathrm{L} \cup \{f\}$
return $\mathsf{KGen}(\mathsf{msk}, f)$

$\underline{\mathrm{Valid}(\mathrm{L}, M_0, M_1)}$:
if $\exists \mathsf{aux}.\exists f \in L : f(M_0, \mathsf{aux}) \neq f(M_1, \mathsf{aux})$
  return 0
return 1

Figure 6.5: The (adaptive) IND-FEAI security experiment, private-key setting.

We define the following properties of an FEAI *scheme:*

- ***Correctness**: for all* $\mathsf{aux} \in \mathcal{X}$ *and for all* $M \in \mathcal{M}$ *the following holds:*

$$\Pr\left[\begin{array}{c|l} \mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{msk}, M), \mathsf{aux}) & \mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \wedge \\ = f(M, \mathsf{aux}) & \mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f) \end{array}\right] \in 1\text{-}\textsc{Negl}(\lambda) \ .$$

- ***Indistinguishability**: for any* PPT *adversary* $\mathcal{A}$*, the following advantage*

$$\mathsf{Adv}^{\mathsf{ind\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) := \left|\Pr[\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda) = 1] - \frac{1}{2}\right|$$

is negligible, where the security experiment $\mathsf{IND\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda)$ *is defined in Figure 6.2.*

6.3 One-Way FEAI and Unbreakability

In the context of FEAI, one-wayness requires that an adversary is not able to recover a randomly sampled $M_1 \in \mathcal{M}_1$ given access to its corresponding ciphertext and to a *single* functional key for function $f : \mathcal{M}_1 \times \mathcal{M}_2 \to \mathcal{Y}$. A clear requirement from f is to be itself a (candidate) one-way function.

Definition 6.3 (One-Way FEAI)**.** *Let* FEAI *stand for a functional encryption with auxiliary inputs scheme in the public-key (private-key) setting. Let* $f : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ *denote a secure pseudorandom permutation. For any* PPT *adversary* $\mathcal{A}$*, the following advantage:*

$$\mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) := \Pr[\mathsf{OW\text{-}FEAI}^{\mathcal{A},f}_{\mathsf{FEAI}}(\lambda) = 1]$$

is negligible, where the security experiment $\mathsf{OW\text{-}FEAI}^{\mathcal{A}}_{\mathsf{FEAI}}(\lambda)$ *is defined in Figure 6.6.*

FEAI achieving one-wayness proves itself sufficiently strong to obtain UBK-secure schemes. We state below such implications.

Lemma 6.2 (OW-FEAI $\Rightarrow$ UBK)**.** *Let* FEAI *be a functional encryption with auxiliary inputs scheme in the public-key setting that is* OW-FEAI*-secure with respect to a pseudorandom permutation* $f : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$*. Then, the advantage of any* PPT *adversary* $\mathcal{A}$ *in winning the* UBK *security experiment against the scheme in Figure 6.7 (top) is bounded as follows:* $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{R},\mathsf{C}_\mathcal{E}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda)$ *. Moreover, if* FEAI *is defined in the private-key setting, the same result holds concerning the scheme in Figure 6.7 such that* $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{R},\mathsf{C}_\mathcal{E}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda)$ *.*

$\underline{\mathsf{OW\text{-}FEAI}^{\mathcal{A},f}_{\mathsf{FEAI}}(\lambda)}$: // Public Setting

$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f)$
$M \leftarrow\!\!\$\ \mathcal{M}$
$C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{mpk}, M)$
$N \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, \mathsf{mpk}, \mathsf{sk}_f, C)$
return $N = M$

$\underline{\mathsf{OW\text{-}FEAI}^{\mathcal{A},f}_{\mathsf{FEAI}}(\lambda)}$: // Private Setting

$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$\mathsf{sk}_f \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{msk}, f)$
$M \leftarrow\!\!\$\ \mathcal{M}$
$C \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{msk}, M)$
$N \leftarrow\!\!\$\ \mathcal{A}(1^\lambda, \mathsf{sk}_f, C)$
return $N = M$

Figure 6.6: The one-wayness security experiment: left for the public-key setting, right for the private-key setting.

$\underline{\mathsf{C.Setup}(1^\lambda, \mathcal{E}, K)}$:

$(\mathsf{msk}, \mathsf{mpk}) \leftarrow\!\!\$\ \mathsf{FEAI.Setup}(1^\lambda)$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathsf{FEAI.KGen}(\mathsf{msk}, \mathcal{E})$
$C_K \leftarrow\!\!\$\ \mathsf{FEAI.Enc}(\mathsf{mpk}, K)$
$\mathsf{C.Eval}^r_k := \mathsf{FEAI.Dec}$
$\mathsf{C.Eval}^r_k.\mathrm{Hardwire}(C_K, \mathsf{sk}_\mathcal{E})$
return $\mathsf{C.Eval}^r_k$

$\underline{\mathsf{C.Eval}^r_k(M)}$:

Hardwire: $\mathsf{sk}_\mathcal{E},\ C_K$
return $\mathsf{FEAI.Dec}(\mathsf{sk}_\mathcal{E}, C_K, M)$

$\underline{\mathsf{C.Setup}(1^\lambda, \mathcal{E}, K)}$:

$\mathsf{msk} \leftarrow\!\!\$\ \mathsf{FEAI.Setup}(1^\lambda)$
$\mathsf{sk}_\mathcal{E} \leftarrow\!\!\$\ \mathsf{FEAI.KGen}(\mathsf{msk}, \mathcal{E})$
$C_K \leftarrow\!\!\$\ \mathsf{FEAI.Enc}(\mathsf{msk}, K)$
$\mathsf{C.Eval}^r_k := \mathsf{FEAI.Dec}$
$\mathsf{C.Eval}^r_k.\mathrm{Hardwire}{:}(C_K, \mathsf{sk}_\mathcal{E})$
return $\mathsf{C.Eval}^r_k$

$\underline{\mathsf{C.Eval}^r_k(M)}$:

Hardwire: $\mathsf{sk}_\mathcal{E}, C_K$
return $\mathsf{FEAI.Dec}(\mathsf{sk}_f, C_K, M)$

Figure 6.7: An UBK symmetric encryption scheme $\mathcal{E}$ based on a functional encryption with auxiliary-inputs scheme FEAI achieving OW-FEAI-security.

Lemma 6.2. For the first part, our reduction $\mathcal{R}$ wins the OW-FEAI game against the underlying FEAI scheme as follows: first, the OW-FEAI game samples M according to the message distribution $\mathcal{M}$ and encrypts it as C. $\mathcal{R}$ receives from the OW-MIFE challenger the functional key associated to $\mathsf{sk}_\mathcal{E}$. After it is provided with $(\mathsf{sk}_\mathcal{E}, C)$, $\mathcal{R}$ constructs the implementation of $\mathcal{E}$. Thus, $\mathcal{A}$ is fed with the implementation which hard-codes two bit-strings $(\mathsf{sk}_\mathcal{E},\ C_k)$. $\mathcal{A}$ – as an adversary for the UBK game against $\mathcal{E}$ – given the implementation of $\mathcal{E}$, extracts the key K with a certain advantage and returns it to $\mathcal{R}$, which forwards it to the OW-FEAI challenger. It is easy to see that $\mathcal{R}$ simulates an implementation of $\mathcal{E}$. If an UBK adversary $\mathcal{A}$ wins, by returning K with a non-negligible, then $\mathcal{R}$ wins the OW-FEAI game with the same advantage. Thus, $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_\mathcal{E}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{R},\mathsf{FEAI}}(\lambda)$.

The second part of the theorem follows in the same manner. Let $\mathcal{A}$ be any PPT adversary able to win the UBK experiment with an advantage $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathcal{E}}(\lambda)$. We construct a reduction $\mathcal{R}$ able to win the OW-FEAI game against the underlying FEAI scheme as follows: After it generates its msk (we are in the private-key setting), the OW-FEAI game samples $M \in \mathcal{M}$ and encrypts it to get the challenge ciphertext C. $\mathcal{R}$ receives from the OW-FEAI challenger the functional key associated to $\mathsf{sk}_\mathcal{E}$. After it is provided with $(\mathsf{sk}_\mathcal{E}, C)$, $\mathcal{R}$ constructs an implementation of $\mathcal{E}.\mathsf{Enc}$. Thus, $\mathcal{A}$ is fed with an implementation which hard-codes two

bit-strings ($\mathsf{sk}_{\mathcal{E}}$, C_K). $\mathcal{A}$ — as an adversary for the UBK game against $\mathcal{E}$ — given an implementation of $\mathcal{E}.\mathsf{Enc}$, extracts the key K with a certain advantage and returns K to $\mathcal{R}$. Finally, K is passed by $\mathcal{R}$ to the OW-FEAI challenger. If an UBK adversary $\mathcal{A}$ wins, by returning K with a non-negligible, then $\mathcal{R}$ wins the OW-FEAI game with the same advantage. Thus: $\mathsf{Adv}^{\mathsf{ubk}}_{\mathcal{A},\mathsf{C}_{\mathcal{E}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{R},\mathsf{FEAI}}(\lambda)$. □

6.4 Relations between OW-MIFE and OW-FEAI

Lemma 6.3 (OW-MIFE ⇒ OW-FEAI)**.** *Let 2in-FE be a two-input functional encryption scheme (public-key setting) achieving* OW-MIFE *with respect to a secure pseudorandom permutation $f : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ and index set $\mathcal{I} = \{1\}$. Then, the* FEAI *construction in Figure 6.4 achieves one-wayness w.r.t. f and $\mathcal{I}$ against any* PPT *adversary $\mathcal{A}$ such that:*

$$\mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 6.3. We construct a reduction $\mathcal{R}$ against the OW-MIFE security of the underlying 2in-FE scheme as follows: the OW-MIFE game samples the pair of keys:

$$(\mathsf{msk}, \mathsf{mpk}_1, \mathsf{mpk}_2) \leftarrow\!\!\$\ \mathsf{2in\text{-}FE.Setup}(1^\lambda),$$

and gives mpk_2 to $\mathcal{R}$. Once $\mathcal{R}$ receives mpk_2, it forwards it to $\mathcal{A}$ as the setting up for FEAI scheme. $\mathcal{R}$ also receives sk_f and sends it back to $\mathcal{A}$. The OW-MIFE game picks uniformly at random the message M_1, encrypts it under the corresponding key, i.e.,

$$C_1 \leftarrow\!\!\$\ \mathsf{2in\text{-}FE.Enc}(\mathsf{mpk}_1, M_1),$$

and sends it to $\mathcal{R}$. The reduction receives and forwards C_1 to $\mathcal{A}$. $\mathcal{A}$ outputs its guess M' and sends it to $\mathcal{R}$ which forwards M' to the experiment. It follows that $\mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \geq \mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda)$. □

A similar result holds in the private-key setting:

Lemma 6.4 (OW-MIFE ⇒ OW-FEAI)**.** *Let* MIFE *be an $(n+1)$-input functional encryption scheme (private-key setting) achieving* OW-MIFE *for $\mathcal{I} = \{1\}$ from Figure 6.8. Then, there exists an* FEAI *construction for $f : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ achieving one-wayness against any* PPT *adversary $\mathcal{A}$:*

$$\mathsf{Adv}^{\mathsf{ow\text{-}feai}}_{\mathcal{A},\mathsf{FEAI}}(\lambda) \leq \mathsf{Adv}^{\mathsf{ow\text{-}mife}}_{\mathcal{R},\mathsf{MIFE}}(\lambda) \ .$$

Lemma 6.4. The proof is straightforward. The reduction $\mathcal{R}$ obtains the challenge ciphertext C_K corresponding to K and then the encryptions of 0s and 1s for the next n positions. $\mathcal{R}$ then simulates the OW-FEAI game ($\mathcal{R}$ also obtains sk_f). If an adversary wins the OW-FEAI game, then $\mathcal{R}$ wins OW-MIFE under the same advantage. □

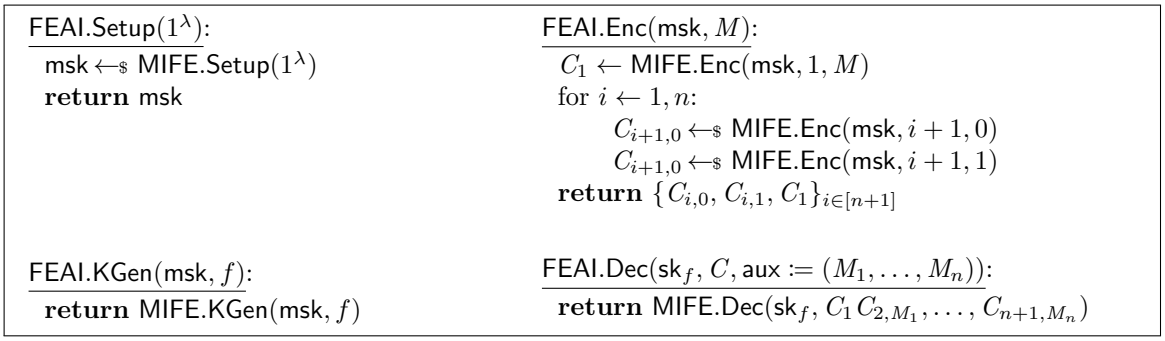

Figure 6.8: The natural construction of FEAI from MIFE (private-key setting).

Chapter 7

Conclusion and Open Questions

This dissertation investigated two main security notions: *robustness* and *unbreakability*. Subsequent work was devoted to achieving them through generic transformations or specific constructions.

Usually, the security notions for symmetric primitives assume that no two ciphertexts issued under different keys collide with noticeable probability. Robustness plays a role in establishing similar guarantees while considering more powerful adversaries, which could even tamper with the key-generation process. Having illustrated the relevance of such questions, we show that obtaining robust authenticated-encryption schemes is realizable via robust and pseudorandom MACs (Section 4.4). In the public-key setting, we extended the recently-introduced notions of strong and complete key-robustness to the settings of digital signature schemes (Section 4.5.1), and functional encryption schemes (Section 3.2.2). While the classical existential unforgeability notion typically guarantees strong robustness, this is not the case for complete robustness — the separation results being accompanied by concrete examples; this motivates the generic transformations of Sections 3.3, 4.4 and 4.5, which result in provable complete-robust schemes.

In Chapters 5 and 6, we looked into the problem of building UBK-secure implementations for block ciphers. In doing so, we put forward the notion of functional encryption with auxiliary inputs (FEAI), which may be of independent interest. We show that FEAI achieving one-wayness is instrumental in providing UBK-compilers for block ciphers, while FEAI reaching indistinguishability is sufficient for obtaining indistinguishability obfuscation (iO). Moreover, we show how to construct such compilers relying on iO or on MIFE achieving one-wayness. To the best of our knowledge, this is the first time the notion of white-box cryptography is shown to be realizable. We hope that our work will shed a new light in developing and deploying provably secure white-box implementations. From a practical point of view, we would like to see implementations based on our ideas or variants thereof, deployed in the field in the future, as a beneficial replacement to current *ad hoc*, heuristic solutions.

7.1 Open Questions

The work presented herein raises some interesting new open problems, and we mention some of them below.

Question 7.1. *Can we build more efficient FE constructions that support general circuits? Would this enable instantiating UBK implementations for block ciphers of lower space complexity and thus lead to efficient/practical solutions?*

Question 7.2. *Can we apply our* UBK*-compiler to other cryptographic primitives such as signatures schemes or* PKE *decryption?*

Question 7.3. *Can FE/MIFE provide similar benefits to other theoretical areas of cryptography, more specifically to program obfuscation?*

Question 7.4. *Can we assess robustness from a practical perspective? Can complete robustness be built-in in the design of primitives (i.e., not through a black-box transform)?*

Question 7.5. *How are the indistinguishability security notions for functional encryption related to the strong robustness notion?*

Question 7.6. *How can the anonymity definition for* FE *schemes be extended to support key-derivation queries?*

Question 7.7. *A different direction would be obtaining functional signature schemes supporting large classes of functionalities. What would be the meaning of robustness for the case of functional signatures?*

Notations

Mathematical Notations

$\mathbb{Z}$	the set of integers
$\mathbb{N}$	the set of non-negative integers
p, p_1, p_2	prime numbers
$(\mathbb{Z}_N, +)$	the additive group over $\mathbb{Z}_N$
$(\mathbb{Z}_N, +, \cdot)$	ring of integers modulo N
$\mathbb{G}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$	cyclic groups
$\mathcal{D} \times \mathcal{R}$	the Cartesian product of $\mathcal{D}$ and $\mathcal{R}$

Algorithmic Concepts

$\{0,1\}^*$	the set of all bitstrings
$\{0,1\}^n$	the set of all bitstrings of length n
$\lvert M \rvert$	length of M
$x \leftarrow y$	assignment
$x = y$	equality
$x \leftarrow\!\!\$\ \mathcal{S}$	x is sampled uniformly at random over the set $\mathcal{S}$

Provable Security

λ	Security Parameter
$\mathcal{A}$	Adversary
$\mathcal{R}$	Reduction
$\text{Game}_1, \text{Game}_2$	Games
$\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_i}(\lambda)$	the advantage of adversary $\mathcal{A}$ in game Game_i

Abbreviations

Cryptographic Primitives

H	Hash Function
PRG	Pseudorandom Generator
PRF	Pseudorandom Function
VRF	Verifiable Random Function
DS	Digital Signature
PKE	Public Key Encryption
ABE	Attribute-Based Encryption
FHE	Fully-Homomorphic Encryption
FE	Functional Encryption
MIFE	Multi-Input Functional Encryption
iO	Indistinguishability Obfuscation
BPO	Best Possible Obfuscation
VBB	Virtual Black-Box Obfuscation

Cryptographic Assumptions

RO	Random Oracle
OW	One-Wayness
OWF	One-Way Function
CR	Collision-Resistance
DDH	Decisional Diffie-Hellman
m-DDH	Multiple DDH
LWE	Learning with Errors
RLWE	Ring Learning with Errors

List of Illustrations

Figures

Personal Publications

- *Simple Constructions of Laconic Function Evaluation from Decomposable FE*
  with Louis Goubin and Lorenzo Spignoli,
  (in submission)
- *White-Box Cryptography from One-Way MIFE*
  with Louis Goubin, Pascal Paillier, Matthieur Rivain and Junwei Wang,
  (in submission)
- *Robust Encryption, Extended*
  with Remi Geraud and David Naccache,
  CT-RSA, March 2019
- *Cryptographic Constructions Based on the Rank Assumption*
  IET Journal version (invited from CANS 2018),
  (in submission)
- *Adaptive-Secure VRFs with Shorter Keys from Static Assumptions*
  CANS, September-October 2018
- *Equiprobability: Faster Algorithms for Subset-Sum Variants and the Vertex Cover Problem*
  with Gustavo Banegas and Matthias Minihld,
  AQIS, September 2018
- *Twisting Lattice and Graph Techniques to Compress Transactional Ledgers*
  with Remi Geraud and David Naccache,
  SecureComm, October 2017
- *Security of Symmetric Primitives under Incorrect Usage of Keys*
  with Pooya Farshim and Claudio Orlandi,
  Fast Software Encryption, March 2017

Bibliography

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. "Simple Functional Encryption Schemes for Inner Products". In: *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Jonathan Katz. Vol. 9020. Lecture Notes in Computer Science. Gaithersburg, MD, USA: Springer, Heidelberg, Germany, Mar. 2015, pp. 733–751. DOI: 10.1007/978-3-662-46447-2_33 (cit. on pp. 39, 45).

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. "Robust Encryption". In: *TCC 2010: 7th Theory of Cryptography Conference*. Ed. by Daniele Micciancio. Vol. 5978. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, Heidelberg, Germany, Feb. 2010, pp. 480–497. DOI: 10.1007/978-3-642-11799-2_28 (cit. on pp. 38, 39, 41, 42, 59, 63, 67, 73, 76, 79).

[ABN18] Michel Abdalla, Mihir Bellare, and Gregory Neven. "Robust Encryption". In: *Journal of Cryptology* 31.2 (Apr. 2018), pp. 307–350. DOI: 10.1007/s00145-017-9258-8 (cit. on pp. 38, 79).

[AJ15] Prabhanjan Ananth and Abhishek Jain. "Indistinguishability Obfuscation from Compact Functional Encryption". In: *Advances in Cryptology – CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2015, pp. 308–326. DOI: 10.1007/978-3-662-47989-6_15 (cit. on p. 32).

[BB04] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles". In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 56–73. DOI: 10.1007/978-3-540-24676-3_4 (cit. on p. 59).

[BB08] Dan Boneh and Xavier Boyen. "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups". In: *Journal of Cryptology* 21.2 (Apr. 2008), pp. 149–177. DOI: 10.1007/s00145-007-9005-7 (cit. on pp. 36, 78).

[BCP02] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. "Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions". In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 2002, pp. 321–336. DOI: 10.1007/3-540-46035-7_21 (cit. on pp. 35, 54).

[BD09] Mihir Bellare and Shanshan Duan. *Partial Signatures and their Applications*. Cryptology ePrint Archive, Report 2009/336. http://eprint.iacr.org/2009/336. 2009 (cit. on p. 76).

[Ber14] Daniel J. Bernstein. *CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness*. https://competitions.cr.yp.to/caesar.html. 2014 (cit. on p. 71).

[BF01] Dan Boneh and Matthew K. Franklin. "Identity-Based Encryption from the Weil Pairing". In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 213–229. DOI: 10.1007/3-540-44647-8_13 (cit. on pp. 14, 40).

[BFLS10] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. "Unlinkability of Sanitizable Signatures". In: *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Phong Q. Nguyen and David Pointcheval. Vol. 6056. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, May 2010, pp. 444–461. DOI: 10.1007/978-3-642-13013-7_26 (cit. on p. 62).

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 1–18. DOI: 10.1007/3-540-44647-8_1 (cit. on pp. 17, 82, 100, 110).

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions". In: *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Buenos Aires, Argentina: Springer, Heidelberg, Germany, Mar. 2014, pp. 501–519. DOI: 10.1007/978-3-642-54631-0_29 (cit. on p. 38).

[Bih94] Eli Biham. "New Types of Cryptanalytic Attacks Using related Keys (Extended Abstract)". In: *Advances in Cryptology – EUROCRYPT'93*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Lofthus, Norway: Springer, Heidelberg, Germany, May 1994, pp. 398–409. DOI: 10.1007/3-540-48285-7_34 (cit. on p. 58).

[BK03] Mihir Bellare and Tadayoshi Kohno. "A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications". In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, May 2003, pp. 491–506. DOI: 10.1007/3-540-39200-9_31 (cit. on p. 58).

[BKS16] Zvika Brakerski, Ilan Komargodski, and Gil Segev. "Multi-input Functional Encryption in the Private-Key Setting: Stronger Security from Weaker Assumptions". In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 852–880. DOI: 10.1007/978-3-662-49896-5_30 (cit. on pp. 19, 82, 83, 91, 95, 110).

[BN08] Mihir Bellare and Chanathip Namprempre. “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm”. In: *Journal of Cryptology* 21.4 (Oct. 2008), pp. 469–491. DOI: 10.1007/s00145-008-9026-x (cit. on p. 62).

[BR93] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols”. In: *ACM CCS 93: 1st Conference on Computer and Communications Security*. Ed. by V. Ashby. Fairfax, Virginia, USA: ACM Press, Nov. 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on p. 27).

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. “Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV”. In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2002, pp. 320–335. DOI: 10.1007/3-540-45708-9_21 (cit. on p. 58).

[BS15] Zvika Brakerski and Gil Segev. “Function-Private Functional Encryption in the Private-Key Setting”. In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 306–324. DOI: 10.1007/978-3-662-46497-7_12 (cit. on pp. 83, 91, 93, 94).

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. “Functional Encryption: Definitions and Challenges”. In: *TCC 2011: 8th Theory of Cryptography Conference*. Ed. by Yuval Ishai. Vol. 6597. Lecture Notes in Computer Science. Providence, RI, USA: Springer, Heidelberg, Germany, Mar. 2011, pp. 253–273. DOI: 10.1007/978-3-642-19571-6_16 (cit. on pp. 14, 32, 38, 40, 43).

[BV15] Nir Bitansky and Vinod Vaikuntanathan. “Indistinguishability Obfuscation from Functional Encryption”. In: *56th Annual Symposium on Foundations of Computer Science*. Ed. by Venkatesan Guruswami. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2015, pp. 171–190. DOI: 10.1109/FOCS.2015.20 (cit. on p. 32).

[CEJv03] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. “White-Box Cryptography and an AES Implementation”. In: *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Kaisa Nyberg and Howard M. Heys. Vol. 2595. Lecture Notes in Computer Science. St. John’s, Newfoundland, Canada: Springer, Heidelberg, Germany, Aug. 2003, pp. 250–270. DOI: 10.1007/3-540-36492-7_17 (cit. on p. 17).

[Chi15] Alessandro Chiesa. *Indistinguishability Obfuscation (Lecture Notes 23)*. 2015. URL: howpublishedttps://people.eecs.berkeley.edu/~alexch/docs/CS276-F2015/lecture-23.pdf (cit. on pp. 100, 101).

[Chu36] Alonzo Church. “An unsolvable problem of elementary number theory”. In: *American journal of mathematics* 58.2 (1936), pp. 345–363 (cit. on p. 13).

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. 2009 (cit. on p. 25).

[CLT15] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. "New Multilinear Maps Over the Integers". In: *Advances in Cryptology – CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2015, pp. 267–286. DOI: 10.1007/978-3-662-47989-6_13 (cit. on p. 17).

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. "Perfectly One-Way Probabilistic Hash Functions (Preliminary Version)". In: *30th Annual ACM Symposium on Theory of Computing*. Dallas, TX, USA: ACM Press, May 1998, pp. 131–140. DOI: 10.1145/276698.276721 (cit. on pp. 36, 53–55).

[CO15] Tung Chou and Claudio Orlandi. "The Simplest Protocol for Oblivious Transfer". In: *Progress in Cryptology - LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America*. Ed. by Kristin E. Lauter and Francisco Rodríguez-Henríquez. Vol. 9230. Lecture Notes in Computer Science. Guadalajara, Mexico: Springer, Heidelberg, Germany, Aug. 2015, pp. 40–58. DOI: 10.1007/978-3-319-22174-8_3 (cit. on pp. 38, 59).

[CS98] Ronald Cramer and Victor Shoup. "A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack". In: *Advances in Cryptology – CRYPTO'98*. Ed. by Hugo Krawczyk. Vol. 1462. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1998, pp. 13–25. DOI: 10.1007/BFb0055717 (cit. on p. 42).

[Dam88] Ivan Damgård. "Collision Free Hash Functions and Public Key Signature Schemes". In: *Advances in Cryptology – EUROCRYPT'87*. Ed. by David Chaum and Wyn L. Price. Vol. 304. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 1988, pp. 203–216. DOI: 10.1007/3-540-39118-5_19 (cit. on p. 79).

[DCW13] Changyu Dong, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol". In: *ACM CCS 13: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Berlin, Germany: ACM Press, Nov. 2013, pp. 789–800. DOI: 10.1145/2508859.2516701 (cit. on p. 59).

[DH76] Whitfield Diffie and Martin E. Hellman. "New Directions in Cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on p. 14).

[DLPR14] Cécile Delerablée, Tancrède Lepoint, Pascal Paillier, and Matthieu Rivain. "White-Box Security Notions for Symmetric Encryption Schemes". In: *SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography*. Ed. by Tanja Lange, Kristin Lauter, and Petr Lisonek. Vol. 8282. Lecture Notes in Computer Science. Burnaby, BC, Canada: Springer, Heidelberg, Germany, Aug. 2014, pp. 247–264. DOI: 10.1007/978-3-662-43414-7_13 (cit. on pp. 17, 82, 84, 110).

[Dod05] Yevgeniy Dodis. "On extractors, error-correction and hiding all partial information". In: *IEEE Information Theory Workshop on Theory and Practice in Information-Theoretic Security, 2005*. IEEE. 2005, pp. 74–79 (cit. on p. 55).

[DS05] Yevgeniy Dodis and Adam Smith. "Correcting errors without leaking partial information". In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 654–663. DOI: 10.1145/1060590.1060688 (cit. on pp. 55, 56).

[Fis99] Marc Fischlin. "Pseudorandom Function Tribe Ensembles Based on One-Way Permutations: Improvements and Applications". In: *Advances in Cryptology – EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, Heidelberg, Germany, May 1999, pp. 432–445. DOI: 10.1007/3-540-48910-X_30 (cit. on pp. 53, 54).

[FLP14] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. "Robust Multi-Property Combiners for Hash Functions". In: *Journal of Cryptology* 27.3 (July 2014), pp. 397–428. DOI: 10.1007/s00145-013-9148-7 (cit. on p. 53).

[FLPQ13] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. "Robust Encryption, Revisited". In: *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Nara, Japan: Springer, Heidelberg, Germany, Feb. 2013, pp. 352–368. DOI: 10.1007/978-3-642-36362-7_22 (cit. on pp. 38, 40–43, 46, 59, 63–65, 73, 76).

[FOR17] Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. "Security of Symmetric Primitives under Incorrect Usage of Keys". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 449–473. ISSN: 2519-173X. DOI: 10.13154/tosc.v2017.i1.449-473 (cit. on pp. 18, 38, 40, 76).

[Gen09] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *41st Annual ACM Symposium on Theory of Computing*. Ed. by Michael Mitzenmacher. Bethesda, MD, USA: ACM Press, May 2009, pp. 169–178. DOI: 10.1145/1536414.1536440 (cit. on p. 30).

[GGG+14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. "Multi-input Functional Encryption". In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Copenhagen, Denmark: Springer, Heidelberg, Germany, May 2014, pp. 578–602. DOI: 10.1007/978-3-642-55220-5_32 (cit. on pp. 82, 85–87, 101, 110).

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th Annual Symposium on Foundations of Computer Science*. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2013, pp. 40–49. DOI: 10.1109/FOCS.2013.13 (cit. on p. 17).

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions". In: *Journal of the ACM* 33.4 (Oct. 1986), pp. 792–807 (cit. on p. 54).

[GJO16] Vipul Goyal, Aayush Jain, and Adam O'Neill. "Multi-input Functional Encryption with Unbounded-Message Security". In: *Advances in Cryptology – ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. Lecture Notes in Computer Science. Hanoi, Vietnam: Springer, Heidelberg, Germany, Dec. 2016, pp. 531–556. DOI: 10.1007/978-3-662-53890-6_18 (cit. on pp. 19, 82, 83, 101–103, 110).

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. "Reusable garbled circuits and succinct functional encryption". In: *45th Annual ACM Symposium on Theory of Computing*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. Palo Alto, CA, USA: ACM Press, June 2013, pp. 555–564. DOI: 10.1145/2488608.2488678 (cit. on pp. 19, 32–34, 82, 83, 91–93, 98, 99, 110).

[GL89] Oded Goldreich and Leonid A. Levin. "A Hard-Core Predicate for all One-Way Functions". In: *21st Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA: ACM Press, May 1989, pp. 25–32. DOI: 10.1145/73007.73010 (cit. on p. 74).

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. "Message Franking via Committing Authenticated Encryption". In: *Advances in Cryptology – CRYPTO 2017, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 66–97. DOI: 10.1007/978-3-319-63697-9_3 (cit. on p. 38).

[GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A "Paradoxical" Solution to the Signature Problem (Abstract) (Impromptu Talk)". In: *Advances in Cryptology – CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1984, p. 467 (cit. on p. 38).

[GNR17] Rémi Géraud, David Naccache, and Răzvan Roşie. "Twisting Lattice and Graph Techniques to Compress Transactional Ledgers". In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2017, pp. 108–127 (cit. on p. 20).

[GNR19] Rémi Géraud, David Naccache, and Razvan Rosie. "Robust Encryption, Extended". In: *Topics in Cryptology – CT-RSA 2019*. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2019, pp. 149–168. DOI: 10.1007/978-3-030-12612-4_8 (cit. on pp. 18, 40).

[GPR+19] Louis Goubin, Pascal Paillier, Matthieu Rivain, Razvan Rosie, and Junwei Wang. "What Functional Encryption and Indistinguishability Obfuscation say to White Box Cryptography". In: (2019) (cit. on pp. 18, 19).

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data". In: *ACM CCS 06: 13th Conference on Computer and Communications Security*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. Available as Cryptology ePrint Archive Report 2006/309. Alexandria, Virginia, USA: ACM Press, Oct. 2006, pp. 89–98. DOI: 10.1145/1180405.1180418 (cit. on pp. 14, 32).

[GR07] Shafi Goldwasser and Guy N. Rothblum. “On Best-Possible Obfuscation”. In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 194–213. DOI: 10.1007/978-3-540-70936-7_11 (cit. on pp. 100, 101).

[HK07] Dennis Hofheinz and Eike Kiltz. “Secure Hybrid Encryption from Weakened Key Encapsulation”. In: *Advances in Cryptology – CRYPTO 2007*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2007, pp. 553–571. DOI: 10.1007/978-3-540-74143-5_31 (cit. on p. 62).

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. “Robust Authenticated-Encryption AEZ and the Problem That It Solves”. In: *Advances in Cryptology – EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer, Heidelberg, Germany, Apr. 2015, pp. 15–44. DOI: 10.1007/978-3-662-46800-5_2 (cit. on p. 58).

[Hor15] Máté Horváth. *Survey on Cryptographic Obfuscation*. Cryptology ePrint Archive, Report 2015/412. http://eprint.iacr.org/2015/412. 2015 (cit. on p. 17).

[HS09] Nadia Heninger and Hovav Shacham. “Reconstructing RSA Private Keys from Random Key Bits”. In: *Advances in Cryptology – CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2009, pp. 1–17. DOI: 10.1007/978-3-642-03356-8_1 (cit. on pp. 40, 76).

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. “OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre-computation Attacks”. In: Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2018, pp. 456–486. DOI: 10.1007/978-3-319-78372-7_15 (cit. on p. 38).

[JNPS16] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. *Deoxys v1.4*. CAESAR candidate. 2016 (cit. on p. 72).

[Kar72] Richard M Karp. “Reducibility among combinatorial problems”. In: *Complexity of computer computations*. Springer, 1972, pp. 85–103 (cit. on p. 15).

[KS17] Ilan Komargodski and Gil Segev. “From Minicrypt to Obfustopia via Private-Key Functional Encryption”. In: *Advances in Cryptology – EUROCRYPT 2017, Part I*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10210. Lecture Notes in Computer Science. Paris, France: Springer, Heidelberg, Germany, May 2017, pp. 122–151. DOI: 10.1007/978-3-319-56620-7_5 (cit. on pp. 33, 82, 91, 95, 100, 110).

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. “Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products”. In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, Heidelberg, Germany, Apr. 2008, pp. 146–162. DOI: 10.1007/978-3-540-78967-3_9 (cit. on p. 14).

[Lam16] Mikkel Lambaek. *Breaking and Fixing Private Set Intersection Protocols*. Cryptology ePrint Archive, Report 2016/665. http://eprint.iacr.org/2016/665. 2016 (cit. on p. 59).

[Lin17] Huijia Lin. "Indistinguishability Obfuscation from SXDH on 5-Linear Maps and Locality-5 PRGs". In: *Advances in Cryptology – CRYPTO 2017, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2017, pp. 599–629. DOI: 10.1007/978-3-319-63688-7_20 (cit. on p. 34).

[LK14] Yehuda Lindell and Jonathan Katz. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014 (cit. on p. 27).

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera: Springer, Heidelberg, Germany, May 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1 (cit. on p. 36).

[LR86] Michael Luby and Charles Rackoff. "How to Construct Pseudo-Random Permutations from Pseudo-Random Functions (Abstract)". In: *Advances in Cryptology – CRYPTO'85*. Ed. by Hugh C. Williams. Vol. 218. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1986, p. 447. DOI: 10.1007/3-540-39799-X_34 (cit. on p. 97).

[Moh10] Payman Mohassel. "A Closer Look at Anonymity and Robustness in Encryption Schemes". In: *Advances in Cryptology – ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Singapore: Springer, Heidelberg, Germany, Dec. 2010, pp. 501–518. DOI: 10.1007/978-3-642-17373-8_29 (cit. on pp. 38, 59).

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. "Verifiable Random Functions". In: *40th Annual Symposium on Foundations of Computer Science*. New York, NY, USA: IEEE Computer Society Press, Oct. 1999, pp. 120–130. DOI: 10.1109/SFFCS.1999.814584 (cit. on p. 78).

[NY90] Moni Naor and Moti Yung. "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks". In: *22nd Annual ACM Symposium on Theory of Computing*. Baltimore, MD, USA: ACM Press, May 1990, pp. 427–437. DOI: 10.1145/100216.100273 (cit. on p. 94).

[ONe10a] Adam O'Neill. *Definitional Issues in Functional Encryption*. Cryptology ePrint Archive, Report 2010/556. http://eprint.iacr.org/2010/556. 2010 (cit. on pp. 38, 43).

[ONe10b] Adam O'Neill. *Deterministic Public-Key Encryption Revisited*. Cryptology ePrint Archive, Report 2010/533. http://eprint.iacr.org/2010/533. 2010 (cit. on pp. 14, 32, 40).

[Pai99] Pascal Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". In: *Advances in Cryptology – EUROCRYPT'99*. Ed. by Jacques Stern. Vol. 1592. Lecture Notes in Computer Science. Prague, Czech Republic: Springer, Heidelberg, Germany, May 1999, pp. 223–238. DOI: 10.1007/3-540-48910-X_16 (cit. on p. 14).

[PS16] Thomas Peyrin and Yannick Seurin. "Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers". In: *Advances in Cryptology – CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 33–63. DOI: 10.1007/978-3-662-53018-4_2 (cit. on p. 72).

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. "On data banks and privacy homomorphisms". In: (1978) (cit. on p. 30).

[RBB03] Phillip Rogaway, Mihir Bellare, and John Black. "OCB: A block-cipher mode of operation for efficient authenticated encryption". In: *ACM Transactions on Information and System Security (TISSEC)* 6.3 (2003), pp. 365–403 (cit. on p. 71).

[Reg05] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *37th Annual ACM Symposium on Theory of Computing*. Ed. by Harold N. Gabow and Ronald Fagin. Baltimore, MA, USA: ACM Press, May 2005, pp. 84–93. DOI: 10.1145/1060590.1060603 (cit. on p. 36).

[Rog02] Phillip Rogaway. "Authenticated-Encryption With Associated-Data". In: *ACM CCS 02: 9th Conference on Computer and Communications Security*. Ed. by Vijayalakshmi Atluri. Washington D.C., USA: ACM Press, Nov. 2002, pp. 98–107. DOI: 10.1145/586110.586125 (cit. on p. 58).

[Ros18] Razvan Rosie. "Adaptive-Secure VRFs with Shorter Keys from Static Assumptions". In: *CANS 18: 17th International Conference on Cryptology and Network Security*. Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2018, pp. 440–459. DOI: 10.1007/978-3-030-00434-7_22 (cit. on p. 20).

[RS06] Phillip Rogaway and Thomas Shrimpton. "A Provable-Security Treatment of the Key-Wrap Problem". In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer, Heidelberg, Germany, May 2006, pp. 373–390. DOI: 10.1007/11761679_23 (cit. on pp. 58, 62).

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. "A Method for Obtaining Digital Signature and Public-Key Cryptosystems". In: *Communications of the Association for Computing Machinery* 21.2 (1978), pp. 120–126 (cit. on p. 14).

[Sak00] Kazue Sako. "An Auction Protocol Which Hides Bids of Losers". In: *PKC 2000: 3rd International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Hideki Imai and Yuliang Zheng. Vol. 1751. Lecture Notes in Computer Science. Melbourne, Victoria, Australia: Springer, Heidelberg, Germany, Jan. 2000, pp. 422–432. DOI: 10.1007/978-3-540-46588-1_28 (cit. on p. 41).

[Sha84] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Advances in Cryptology – CRYPTO'84*. Ed. by G. R. Blakley and David Chaum. Vol. 196. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1984, pp. 47–53 (cit. on pp. 14, 40).

[Sin99] Simon Singh. "The Alternative History of Public-Key Cryptography". In: (1999). https://cryptome.org/ukpk-alt.htm (cit. on pp. 13, 14).

[SW05] Amit Sahai and Brent R. Waters. "Fuzzy Identity-Based Encryption". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Aarhus, Denmark: Springer, Heidelberg, Germany, May 2005, pp. 457–473. DOI: 10.1007/11426639_27 (cit. on p. 40).

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *46th Annual ACM Symposium on Theory of Computing*. Ed. by David B. Shmoys. New York, NY, USA: ACM Press, May 2014, pp. 475–484. DOI: 10.1145/2591796.2591825 (cit. on pp. 17, 28, 110).

[Tur37] Alan M Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London mathematical society* 2.1 (1937), pp. 230–265 (cit. on p. 25).

[Wat11] Brent Waters. "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization". In: *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi. Vol. 6571. Lecture Notes in Computer Science. Taormina, Italy: Springer, Heidelberg, Germany, Mar. 2011, pp. 53–70. DOI: 10.1007/978-3-642-19379-8_4 (cit. on p. 14).

[Yao82] Andrew Chi-Chih Yao. "Theory and Applications of Trapdoor Functions (Extended Abstract)". In: *23rd Annual Symposium on Foundations of Computer Science*. Chicago, Illinois: IEEE Computer Society Press, Nov. 1982, pp. 80–91. DOI: 10.1109/SFCS.1982.45 (cit. on p. 74).

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets (Extended Abstract)". In: *27th Annual Symposium on Foundations of Computer Science*. Toronto, Ontario, Canada: IEEE Computer Society Press, Oct. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25 (cit. on p. 31).

RÉSUMÉ

Cette thèse s'intéresse à la faisabilité des implémentations en boîte blanche de permutations pseudo-aléatoires sûres. Concrètement nous montrons comment un schéma de chiffrement fonctionnel à plusieurs entrées, qui satisfait une notion naturelle d'être à sens unique, est fondamental à la construction d'implémentations protégées contre les attaques d'extraction de clés.
Comme contribution indépendante possédant son intérêt propre, nous étendons la notion de robustesse cryptographique. Sommairement, le chiffrement robuste garantit qu'un chiffré ne peut être lu au moyen de plusieurs clés. Décrite tout d'abord dans le contexte de la cryptographie à clé publique, nous étendons les définitions aux contextes du chiffrement fonctionnel et à l'authentification.

MOTS CLÉS

cryptographie en boîte-blanche, robustesse, chiffrement fonctionnel

ABSTRACT

This thesis investigates the realizability of white-box implementations for secure pseudorandom permutations. Concretely, we show that multi-input functional encryption achieving a natural definition of one-wayness is instrumental in building implementations that are secure against key-extraction attacks.
As a contribution of independent interest, we extend the notion of robustness to a larger set of primitives. Roughly speaking, robust encryption guarantees that a ciphertext cannot be decrypted under different keys. Initially formalized in a public-key context, we introduce compelling definitions for authentication and functional encryption schemes.

KEYWORDS

white-box cryptography, robustness, functional encryption