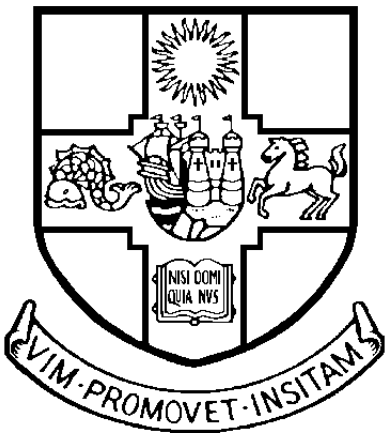
Extensions of Public-Key, Identity-Based and Certificateless Encryption Schemes

Pooya Farshim



A dissertation submitted to the University of Bristol in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Engineering, Department of Computer Science.

February 2008

Abstract

In this work, we first study extensions and applications of the identity-based encryption (IBE) primitive, focusing on generic constructions. We first define what is meant by an identity-based key encapsulation mechanism before extending it to the multi-recipient setting. We then propose an efficient multi-recipient scheme based on bilinear maps and then move on to investigate the properties of public-key schemes which allow for generic and efficient construction of multi-recipient encryption schemes via randomness-reuse. We conclude this part by proposing a generic construction of workflow cryptosystems, where encryption is performed with respect to an access structure, based on any IBE scheme.

We then turn our attention to the certificateless encryption primitive and extend it to the hybrid encryption paradigm. Subsequently, a certificateless key encapsulation mechanism based on weakly secure identity-based and public-key encryption schemes is constructed. Next, by introducing the first provably secure certificateless signcryption scheme, we demonstrate how one can achieve two of the most important security goals in cryptography, namely confidentiality and authenticity (including non-repudiation) of data, efficiently in the certificateless scenario.

Acknowledgements

Firstly, I would like to thank my supervisor Nigel Smart. His enthusiasm, energy and encouragement have been a constant source of inspiration.

I was fortunate enough to have met Manuel Barbosa and spent two excellent months during my studies with him at Departamento de Informática, Universidade do Minho in Portugal. We have had many lively conversations over the past three years. He has been an excellent friend and a great intellectual companion. I would not have enjoyed my PhD as much without his collaboration.

I also would like to offer my thanks to John Malone-Lee for his collaboration and helpful discussions (including those non-academic) as well as Kenny Paterson and Bogdan Warinschi for providing detailed and insightful comments on this thesis.

I am very grateful to Viresh Patel for many enjoyable internet and occasionally face-to-face conversations during the past years, Gary Batt for joining me for a few half-pints (of Leffe) whenever he could, Paul Morrissey for all the silly-but-necessary jokes we had, as well as Mark McPate and Andrew Woodward for being excellent flatmates. Whenever possible, catching up with friends overseas, in particular with Hamid Khorramdel and Hatice Koyuncu, was a much needed and invigorating experience!

Finally, I would like to thank my mother, Narmin Baraheni, for all her invaluable support throughout. This thesis is dedicated to her.

Contents

Chapter 1

Introduction

1.1 Provable Security

Provable security, a misleading term perhaps[1], is the mathematical study of design and analysis of cryptographic algorithms. It deals with questions such as: What is an *encryption scheme*? What is *meant* by saying that an encryption scheme is secure? How can one *establish* security of a cryptosystem? And what are the *concrete* security guarantees in practice? It tries to answer these questions in a mathematically precise way, departing from the traditional approach of ad hoc analysis, where the system is re-designed after a vulnerability is identified.

To answer the above questions, one therefore formulates the following:

1. A precise definition of the cryptosystem, describing the input/output behaviour of various algorithms (the *primitive*);

2. A formal definition of security which specifies what it means to break the system and what resources the adversary has access to when launching an attack (the *security model*);

[1] As one might consider a notion of unproven security.

3. A concrete instantiation of the primitive (a cryptographic *construction*);

4. A security proof showing the impossibility of existence of an adversary breaking the cryptographic construction with respect to the security model. (a *reduction* when dealing with computational security); and

5. An analysis of the concrete behaviour of the reduction in practice (the *tightness* of reduction).

This modern approach to security is inspired by notions derived from the theory of computational complexity where one, usually, identifies an *efficient* algorithm with a *probabilistic polynomial-time* Turing machine. Let us note that by polynomial-time we mean a running time which is bounded by a polynomial function in the length of the input instance. The behaviour of algorithms is analysed as the length parameter, referred to as the security parameter in cryptography, tends to infinity. One then compares various computational problems via *reductions* between their problem instances. A reduction from a problem A to problem B is an algorithm which can solve the problem instances of A with the help of any algorithm which solves the instances of problem B. If this reduction is efficient, i.e. it runs in polynomial time, and the problem instances of B can be solved efficiently, then problem A is, by definition, also easy to solve.

Put differently, to show the infeasibility of breaking a cryptosystem, one assumes the existence of an efficient adversary (an algorithm) breaking the cryptosystem and then proceeds to build a new algorithm, via an efficient reduction to the adversary, which solves a problem which is deemed infeasible[2]. Such a formal treatment originates from the work of Shannon [81] in 1949, Rabin [74] and more formally and explicitly from that of Goldwasser and Micali [49]. More and more

[2]The infeasibility of the latter problem is often assumed, since an unconditional proof often implies P$\neq$NP, which remains the most important open problem in theoretical computer science.

standardisation organisations are adapting this complexity-theoretic approach for security analysis.

The importance of the tightness of reduction was brought to the attention of practitioners by Bellare though his paper on practice-oriented provable-security [12]. The complexity-theoretic approach to cryptography only gives an asymptotic analysis of security. In practice, however, one needs to use a concrete value for the security parameter. This value should guarantee that any attempt to break the system will take more than a specified number of operations. One therefore analyses the performance of a reduction, i.e. its run-time and probability of success, and chooses an appropriate value for the security parameter by taking into account the performance of the state-of-the-art algorithms for solving the underlying hard problem.

1.2 Secret-Key and Public-Key Encryption

A secret-key, or private-key, or symmetric encryption (SKE) scheme, informally, consists of an algorithm to produce keys $\mathtt{k}$, an encryption algorithm which takes a message $\mathtt{m}$ and a key $\mathtt{k}$ and returns a ciphertext $\mathtt{c}$, and a decryption algorithm which undoes encryption by taking the ciphertext $\mathtt{c}$ and the *same* key $\mathtt{k}$ and returning the message $\mathtt{m}$. An example of an SKE scheme is the so-called *one-time pad* where encryption returns $\mathtt{c} = \mathtt{m} \oplus \mathtt{k}$ and decryption recovers $\mathtt{m}$ by computing $\mathtt{m} = \mathtt{c} \oplus \mathtt{k}$. Here $\oplus$ denotes the bitwise exclusive or operand: for a and b two bits, we define $a \oplus b = 0$ if $a = b$ and 1 otherwise.

Shannon [81] gave a security proof for the above scheme. Consider a message $\mathtt{m} \in \{0,1\}$ being encrypted by taking a key bit $\mathtt{k}$ uniformly at random and returning $\mathtt{m} \oplus \mathtt{k}$. Now suppose an eavesdropping adversary sees a 1 being transmitted over a network. There are two possibilities: either $\mathtt{m} = 1$ and $\mathtt{k} = 0$, or $\mathtt{m} = 0$ and

$\mathtt{k} = 1$. Since the key is selected uniformly at random, both these events are equally likely, and the adversary can draw no conclusions about the original message. If the message is a 0, then the adversary will see a 0 with probability $1/2$ and a 1 with probability $1/2$. Similarly, if the message is a 1, then the adversary will see a 0 with probability $1/2$ and a 1 with probability $1/2$. Hence, the adversary has no *more* information than he would, by mere probabilistic guessing. Note that this argument shows that the one-time pad encryption scheme is *information-theoretically secure*: it is *impossible* to recover the plaintext without the key.

Diffie and Hellman [42] in 1976 introduced the notion of public-key cryptography. In a public-key encryption (PKE) scheme, a widely available public key is used for the encryption of messages and a matching private key is then used to decrypt the resulting ciphertexts. This encryption paradigm is a very useful tool in many situations. For example, it is used to enable communication over the Internet as there is no need, unlike a symmetric scheme, for the parties to agree on a common secret before any messages are exchanged. The first PKE scheme was proposed by Rivest, Shamir and Adelman in [75]. Security of this scheme was argued for on the grounds that the integer factorisation problem is intractable. However it was Rabin [74] who modified their scheme in way which made a reduction to the factorisation problem possible.

Goldwasser and Micali [49] formulated a precise notion of security for PKE schemes, known as semantic security, akin to the information-theoretic definition proposed by Shannon: for a scheme to be semantically secure, it should be *infeasible* (rather than impossible) to find any information about a plaintext from a ciphertext without being in possession of the required private key, except maybe the length of the message.

This notion was, however, quite cumbersome and was later modified to equivalent or stronger alternatives which were more manageable to work with. The

most practical definition, known as indistinguishability of ciphertexts, requires it to be infeasible to distinguish which of the two known messages is encrypted in a ciphertext. Although already a strong notion, this definition does not model the scenario where an adversary could obtain extra information by analysing the decryption behaviour of ciphertexts by, say, submitting them to a user. The model was extended and the notion of *indistinguishability under chosen ciphertext attacks* (IND-CCA) was formulated. This security definition, roughly speaking, requires indistinguishability of ciphertexts even in presence of an *oracle* which decrypts ciphertexts of an adversary's choice, except the ciphertext which it is trying to break. Nowadays, the moral behind a good security model, based on a famous quote of Einstein, is that everything should be made as simple as possible for the adversary, but not simpler. The IND-CCA definition is now accepted as the gold standard of security for encryption in modern cryptography. Its precise definition, and various variations thereof, will be presented in Section 2.3.2.

The RSA encryption scheme was modified further resulting in the RSA-OAEP scheme which was shown to be IND-CCA secure in [46] in a non-standard but widely accepted model, known as the *random oracle model* (ROM), where it is assumed that cryptographic hash functions behave in a perfectly random way (see Section 2.3.1). The first practical and IND-CCA secure encryption scheme in the standard model of computation (i.e. without relying on assumptions such as the random oracle model) was given in a seminal paper by Ronald Cramer and Victor Shoup in [37].

1.3 The KEM-DEM Paradigm

One of the problems associated with public-key cryptography is its relative inefficiency compared to symmetric cryptosystems. This inefficiency is a result of

using algebraic operations, such as a group addition, rather than fast bit operations used in symmetric primitives. Most practical public-key encryption schemes use a *hybrid* combination of the two primitives to handle large plaintexts. Rather than using the public-key encryption algorithm directly over the plaintext, one generates a random *session key* and uses it to encrypt the message under a more efficient symmetric algorithm. The (small) session key is then encrypted using the public-key encryption algorithm, and the message is transferred as the combination of both ciphertexts.

More explicitly, in this approach to hybrid encryption one defines a symmetric *data encapsulation mechanism* (DEM) which takes a key $\mathtt{k}$ and a message $\mathtt{m}$ and computes $\mathtt{c} = \mathtt{DEM}_{\mathtt{k}}(\mathtt{m})$. Given the knowledge of $\mathtt{k}$, one can recover $\mathtt{m}$ via $\mathtt{m} = \mathtt{DEM}_{\mathtt{k}}^{-1}(\mathtt{c})$. The key $\mathtt{k}$ is transferred to the recipient of the ciphertext in the form of an encapsulation. To create this encapsulation, the sender uses a *key encapsulation mechanism* (KEM). This is an algorithm which takes as input a public key $\mathtt{PK}$ and outputs a session key $\mathtt{k}$ plus an encapsulation $\mathtt{c}'$ of this session key under the public key. We write this as

$$(\mathtt{k}, \mathtt{c}') \leftarrow \mathtt{KEM}(\mathtt{PK}).$$

Notice that the session key $\mathtt{k}$ is not passed as an input to the KEM. This feature could improve the efficiency of a KEM compared to a PKE in various ways, the most common being better bandwidth usage due to shorter ciphertexts and choice of smaller security parameter due to a tighter security reduction.

The recipient recovers the key $\mathtt{k}$ using his private key $\mathtt{SK}$ via the decapsulation mechanism. We denote this by

$$\mathtt{k} \leftarrow \mathtt{KEM}^{-1}(\mathtt{c}', \mathtt{SK}).$$

The full ciphertext of the message $\mathtt{m}$ is then given by $(\mathtt{c}', \mathtt{c})$. Interestingly, we

gain an extra functionality from this construction. Suppose a user receives a large ciphertext in an untrustworthy environment. If the secret key SK of the user is compromised in this environment, all his future communication will no longer be private. To avoid this problem, the user can simply transfer the small KEM component of the ciphertext to a server, where his secret key is stored securely. The session key k is then retrieved remotely and transferred back to the user which enables him to obtain the message.

The hybrid public-key encryption paradigm has gained strength in recent years with the definition and formal security analysis of the generic KEM-DEM construction [37, 38, 86]. The use of the KEM-DEM paradigm allows different components of a hybrid encryption scheme to be designed in isolation, leading to a simpler analysis and hopefully more efficient schemes. In fact, Cramer and Shoup (see Section 2.4.1) established that if the KEM and DEM components are individually secure, the resulting hybrid public-key encryption scheme will be also secure. The relations between security notions for KEMs and the necessary and sufficient conditions for the security of a KEM-DEM construction are discussed further in [70] and [52] respectively.

We summarise the main benefits of the KEM-DEM paradigm over direct use of PKE schemes:

- Encryption of large messages is performed significantly faster;
- Messages need not to be encoded as elements of an algebraic space;
- The cryptosystem is designed in a modular way;
- One might be able to present a tighter security reduction; and
- Potentially the construction could offer extra functionality.

1.4 Identity-Based Encryption

Another problem associated with public-key encryption is that public keys are random bit-strings and hence contain no direct semantic meaning to users. Typically, we use a *certificate*, issued by a trusted *certification authority*, to attach a semantic meaning to a public key, such as binding it to a user identity. This leads to the problematic area of certificate management.

There are a number of other paradigms for public-key encryption which avoid some of these problems. The most obvious is identity-based encryption (IBE) initially proposed by Shamir in [80]. In this scenario, users can set their public keys to be any string of their choice, such as their phone number, e-mail address or IP address. Private keys are then derived from their identities via a trusted authority (TA), who possess a master secret key. IBE schemes thus eliminate the need for certificates, as it is the receiver who ensures her private key is obtained securely.

A secure and efficient identity-based encryption scheme, secure in the random oracle model, was introduced by Boneh and Franklin [23] by utilising bilinear maps on elliptic curves. Sakai and Kasahara [77] later[3] proposed another IBE scheme, also based on bilinear maps, but adopting a different key construction. The security of this scheme in the ROM was established by Chen et al. in [30]. Recently Waters [88] and Gentry [48] have proposed practical IBE schemes which are secure in the standard model.

Lynn [66] mentions that the encryption algorithm proposed by Boneh and Franklin is unlikely to be used, since in practice one will use a form of key encapsulation. We call such an encapsulation mechanism an identity-based key encapsulation mechanism (IBKEM). Lynn proceeds to mention a possible IBKEM construction, but he gives no security model or proof. In Chapter 3 we define what is meant by a key encapsulation for the identity-based setting. Having done this, we show

[3] Although their paper appeared later, their scheme dates earlier than that of Boneh and Franklin's.

that Lynn's construction for an IBKEM can be used to build a fully secure identity-based encryption scheme, when combined with an appropriate DEM. The resulting scheme is computationally more efficient than the original Boneh–Franklin construction. The security proof is tighter for Lynn's construction. However, the proof of security relies on a stronger intractability assumption than that used to base the security of Boneh–Franklin scheme on. We also present a generic construction of an IBKEM, which is secure in a strong sense, from any identity-based encryption scheme, which is secure in a weak sense. When we instantiate this generic scheme with BasicIdent from [23], we obtain an IBKEM which is as efficient as the Boneh–Franklin construction, and which is based on the, now standard, bilinear Diffie–Hellman problem. We feel our construction of an identity-based encryption scheme from an IBKEM and a standard DEM is more natural than the construction in [23], which relies on the Fujisaki–Okamoto transform [45]. An IBKEM, which is fully secure in the standard model, is constructed by Kiltz [58] based on the IBE scheme of Waters [88].

1.5 Multi-Recipient Encryption

It is often the case that one needs to transmit a message securely to multiple parties who hold public/private keys corresponding to a standard single-user encryption scheme. This problem was first studied by Kurosawa [60] who proposed several schemes with substantial improvements over the trivial solution of running multiple, independent instantiations of the encryption algorithm. He was able to share the values computed for many users, lowering the bandwidth and computational requirements of the multi-recipient scheme.

Smart [83] extends the KEM primitive to the multi-recipient setting. Given that KEMs internally chose the session key, he considers the question: is it possible, in

this situation, to avoid carrying out n instances of the KEM-DEM construction? He then proposes a multiple KEM (or an mKEM) primitive whereby the sender can create an encapsulation of the session key $\mathtt{k}$ under n public keys. The purpose of this type of construction is to save $n-1$ data encapsulations overall: a single DEM invocation using $\mathtt{k}$ suffices to obtain valid ciphertexts for all recipients. The entire message encryption process becomes:

$$\begin{aligned}(\mathtt{k}, \mathbf{c}) &\leftarrow \mathtt{mKEM}(\mathtt{PK}_1, \ldots, \mathtt{PK}_n)\\ \mathtt{c} &\leftarrow \mathtt{DEM}_{\mathtt{k}}(\mathtt{m}),\end{aligned}$$

and each recipient i will get the pair $(\mathbf{c}, \mathtt{c})$. For decryption, user i simply performs the following two operations:

$$\begin{aligned}\mathtt{k} &\leftarrow \mathtt{mKEM}^{-1}(\mathbf{c}, \mathtt{SK}_i)\\ \mathtt{m} &\leftarrow \mathtt{DEM}_{\mathtt{k}}^{-1}(\mathtt{c}).\end{aligned}$$

One of the contributions of this thesis is to bring together the concepts of IBKEM and mKEM and propose identity-based key encapsulation to multiple users. In Chapter 4, after giving the appropriate primitive and security definitions, we present a secure and efficient multi-recipient scheme based on bilinear maps.

1.6 Randomness-Reuse

Generating randomness for cryptographic applications is a costly and security-critical operation. It is often assumed in security analysis that random coins are sampled from a perfect uniform distribution and are handled securely. Moreover, various operations performed by a cryptographic algorithm depend on these random coins. These operations, such as a group exponentiation, can be quite costly and prevent the use of the scheme in constrained devices. Therefore, minimising the amount of fresh randomness required in cryptographic algorithms is important

for their overall efficiency and security.

One approach to minimise this problem is to *reuse randomness* across multiple instantiations of cryptographic algorithms, namely in the context of batch operations where (possibly different) messages are encrypted to multiple recipients. This avenue must be pursued with caution, since randomness-reuse may hinder the security of cryptographic schemes. However, when possible, this technique allows for significant savings in processing load and bandwidth, since partial results (and even ciphertext elements) can be shared between multiple instances of a cryptographic algorithm.

Examples of this method are the multi-recipient encryption schemes proposed by Kurosawa [60], the mKEM by Smart in [83], the certificateless encryption schemes in [5] and [4] where randomness is shared between the identity-based and public-key components, and the cryptographic workflow scheme (see the next section) in [3]. Let us note that the scheme of [4] was broken and repaired by Libert and Quisquater in [63].

Bellare et al. [14] extend Kurosawa's work in a number of ways. First, they consider the problem of sending many messages, instead of a single message as discussed in Section 1.5, and give a stronger security model for such schemes. They then systematically study the problem of reusing randomness. The authors examine multi-recipient encryption schemes, and consider the particular case of constructing such schemes by running multiple instances of a public-key encryption scheme, whilst sharing randomness across them. An interesting result in this work is a general method for identifying PKE schemes that are secure when used in this scenario. Schemes which satisfy the so-called *reproducibility test* are guaranteed to permit a proof strategy based on a hybrid argument which is generally captured in a *reproducibility theorem*. Bellare et al. [16] later leveraged on these results to propose a stateful encryption framework which enables more efficient

(single-recipient) encryption operations.

In Chapter 5 we extend the above theoretical framework supporting the reuse of randomness to construct multi-recipient encryption schemes. The main contribution of our work is a more permissive test that permits constructing a wider class of efficient *single-message* multi-recipient schemes. Of particular interest are the optimised versions of the ElGamal and Cramer–Shoup multi-recipient encryption schemes briefly mentioned by Kurosawa in the final section of [60]. We show that these schemes do not fit in the randomness-reuse framework originally proposed by Bellare et al. in [14] and propose extensions to the original definition which capture these as well as other similar schemes.

We then turn our attention to the KEM-DEM paradigm and focus on key encapsulation mechanisms. Adaptation of the results in [14] is straightforward if one focuses on multi-recipient KEMs generating independent keys for each recipient. The interesting case arises when one considers single-key multi-recipient KEMs. To construct these schemes efficiently by reusing randomness, we define the notion of *public key independent* KEM. However, we find that if such a KEM satisfies an appropriate modification of the reproducibility test of Bellare et al. [14] it cannot be secure. To compensate for this negative result, we propose an alternative generic construction of efficient single-key multi-recipient KEMs based on weakly secure PKEs which pass our new reproducibility test. This construction generalises the mKEM in [83]. We also propose a concrete efficient construction of single-key multi-recipient KEMs, which can be shown to be IND-CCA secure in the standard model using techniques in [37].

As in Bellare et al.'s work [14], our reproducibility theorem has a linear security degradation in the number of users. In Section 5.4 we discuss a possible solution to this problem by introducing another test for randomness-reuse. We conclude the work on randomness-reuse by briefly considering its extensions to the IBE setting.

1.7 Cryptographic Workflow

Let us now turn to an application of identity-based encryption schemes and see how they can be used to build cryptographic workflow schemes. The term *workflow* is used to describe a system in which actions must be performed in a particular order. In *cryptographic workflow* [72] this is achieved by making decryption a privileged action which can only be executed by users which possess an appropriate set of *authorisation credentials*, or simply *credentials*. Credentials are issued by a set of *authorisation authorities*, which can ensure that some action has been performed, or that some event has occurred, before granting them to users. Restricting access to encrypted messages in this way, workflow mechanisms can be implemented with cryptographic security guarantees.

An encryption scheme supporting cryptographic workflow should provide the following functionality [3]. Alice specifies the credentials that Bob should have in a *policy* that she decides before encrypting. Alice should be able to perform this encryption without knowing what credentials Bob actually has. A particular authorisation authority will validate that Bob is entitled to a given credential before awarding it. Each credential acts as a partial decryption key. Alice may also want to be sure that no colluding set of these authorisation authorities is able to decrypt and recover the message that she intended for Bob. If this is the case, the system should be *escrow-free*.

Cryptographic workflow follows from the original ideas by Chen et al. in [32, 33]. There the authors explored the possibilities of using Boneh and Franklin's IBE scheme in a setting where a user can extract different identity-based private keys from multiple TAs. They proposed using *credential descriptors* as public keys, in place of the usual identity strings, and showed that combining the master public keys of the TAs in different ways, one may securely send a message to a recipient and restrict her ability to decrypt it with a high degree of flexibility. Smart [82]

applied the same principle to access control. Paterson [72] first employed the term *workflow* to describe this type of scheme.

Al-Riyami et al. [3] formalised the definitions of primitives and security models associated with cryptographic workflow and proposed an efficient escrow-free encryption scheme supporting cryptographic workflow. The scheme is based on Boneh and Franklin's IBE scheme and is proved secure under two security notions. The first one, called *receiver security*, ensures that only users with an appropriate set of credentials can decrypt the message. The second, called *external security*, captures the escrow-freeness notion: it must be unfeasible for any colluding set of TAs to decrypt the message. Unlike certificateless public-key cryptosystems, escrow-freeness is achieved using a classical public-key encryption layer which relies on public key certification.

Encryption schemes supporting cryptographic workflow are very close to those associated with hidden credential systems [27, 53]. Both types of schemes typically employ a secret sharing layer and an identity-based encryption layer, although the goals in each case are different. In hidden credential systems one seeks to keep the access control policy secret, whereas in workflow schemes this is not the case. Workflow cryptosystems are also related to attribute based encryption schemes [76, 50] where one selectively shares encrypted data.

In Chapter 6 we introduce the notion of KEMs supporting cryptographic workflow (WF-KEM) and their escrow-free counterparts (EFWF-KEM). We adapt the security models proposed in [3] for encryption schemes accordingly. We argue that the KEM-DEM paradigm also applies when one moves to encryption schemes supporting cryptographic workflow. In fact, we show that combining a secure WF-KEM (EFWF-KEM) with a secure DEM, results in a secure (escrow-free) encryption scheme supporting cryptographic workflow.

We then present a generic construction that permits building WF-KEMs out

of simpler cryptographic primitives. This is a generalisation of the construction presented in [3] based on the identity-based encryption scheme of Boneh and Franklin. We show how one can construct analogous schemes by replacing its building-blocks with other components providing the same functionality. More specifically, we prove that our transformation permits constructing a secure WF-KEM using secure IBE and secret sharing schemes [79, 19, 59]. Finally, we extend our generic construction to obtain an EFWF-KEM using a secure public-key encryption scheme. Chosen ciphertext security is achieved via a one-time signature scheme (see Section 2.3.7). Our constructions are all secure in the standard model.

A common feature of many schemes proposed for certificateless cryptosystems, cryptographic workflow and hidden credentials is that they are based on the concept of multiple encryption (or re-encryption). In multiple encryption, a ciphertext is created by combining the results of several instances of an encryption algorithm with different encryption keys. In the simplest case, where only two decryption keys are involved, the objective is that even if the adversary is in possession of one of those keys, she obtains no advantage. Recently, Dodis and Katz [43] have addressed the chosen ciphertext security of multiple encryptions in the general case, and have proposed generic constructions which are semantically secure. Our constructions build on these results.

1.8 Certificateless Encryption

Although IBE schemes simplify key management problems by eliminating the need for certificates, this advantage comes at a price: IBE schemes are inherently non-escrow-free as the trusted authority can decrypt any ciphertext. Certificateless encryption (CLE) schemes eliminate this escrow property while preserving the certificateless nature of the system. This form of encryption was introduced

and developed in a series of works by Al-Riyami and Paterson [5, 2, 4].

Roughly speaking, in a CLE scheme, a user's full private key is a combination of a locally computed secret value and a partial private key received from a Key Generation Centre (KGC). This means an adversary can launch two types of attacks: (1) either replace the public key of the user (due to lack of certificates) or (2) act as a malicious KGC and have access to all user partial private keys. In each scenario, if either the partial private key or the secret value remains unexposed, the privacy of data should remain intact.

Al-Ryiami and Paterson [5] proposed a CLE scheme based on the IBE scheme of Boneh and Franklin. The encryption scheme was proven secure under an unrealistic security model where an adversary may replace users' public keys and demand decryptions without providing the corresponding secret value (see the last remark in Section 2.3.6). Other secure constructions of CLE schemes based on the IBE schemes of Boneh and Franklin and that of Sakai and Kasahara [77] are given in [84] and [35] respectively. Recently, Dent et al. [41] have proposed certificateless encryption schemes, including a practical construction based on the IBE scheme of Waters [88], which provide strong security guarantees in the standard model. In [63], Libert and Quisquater generalised the encryption scheme in [5] by proposing a generic construction of certificateless encryption schemes from identity-based and public-key encryption in the random oracle model. A good survey of certificateless public-key encryption schemes and security models can be found in [39].

A contribution of this thesis is to present a security model for key encapsulation to build certificateless encryption (CLKEM). We show in Chapter 7 that combining a secure CLKEM with a secure DEM results in a secure certificateless encryption scheme. In order to prove our composition result we need to modify the unrealistic definitions of security for certificateless schemes to one where decryption is per-

formed on replaced public keys if their matching secret value is provided. We will discuss this point further once we have introduced the appropriate security notions.

We will also describe a generic construction of strongly secure CLKEMs based on any (weakly secure) identity-based encryption scheme and a special form of (weakly secure) public-key encryption scheme, such as RSA or ElGamal in certain groups. This generic approach allows one to add certificateless encryption onto an infrastructure of existing RSA and ElGamal keys, which are either not certified or whose certificates are not trusted by the sender.

1.9 Certificateless Signcryption

Signcryption, a cryptographic primitive introduced by Zheng in [91], captures a common practical scenario where one simultaneously requires confidentiality, authenticity and non-repudiation of transmitted data. Ideally, this should allow for improvements in the overall security and efficiency of the resulting cryptosystems over independent use of an encryption and a digital signature scheme. The security goals associated with signcryption are stronger than those provided by authenticated encryption, where data authenticity suffices and non-repudiation is not required. Authenticity ensures the recipient of the origin and integrity of the data whereas non-repudiation makes it impossible for the sender to deny creating the signcrypted data. For this reason the security model for signcryption should capture *insider attacks* where a dishonest receiver should not be able to forge a valid signcryption originating from another user. This ensures non-repudiation is in place: the receiver can prove the origin of data by providing its secret de-signcryption key. In less common scenarios one may also require forward secrecy, where a message sent by a legitimate user, cannot be decrypted even by an adversary which later is able to get hold of the sender's secret key.

1.9 Certificateless Signcryption

This primitive has been extensively studied in the public-key and identity-based settings where many efficient and secure schemes have been proposed. A systematic study of the properties of the signcryption schemes resulting from the black-box composition of encryption and signature schemes was presented by An et al. [6]. More efficient generic constructions of signcryption schemes based on Tag-KEMs were introduced by Bjørstad and Dent [21]. Malone-Lee [68] studied signcryption with non-interactive non-repudiation and formulated signcryption in the identity-based setting [67]. This work was extended by Boyen [26] who proposed a more structured primitive definition and comprehensive security notions which captured different security guarantees that could be achieved by an identity-based signcryption scheme. Other efficient constructions were subsequently proposed by Libert and Quisquater in [62], and Chen et al. in [34].

Since certificateless cryptography was introduced by Al-Ryiami and Paterson [5], numerous certificateless encryption and signature schemes, and variants thereof, have been proposed. However, the equivalent of public-key signcryption has not been considered in the certificateless setting. In Chapter 8 we introduce the notion of certificateless signcryption, define appropriate security models, and propose an efficient scheme which is provably secure against insider attacks in the random oracle model. Our certificateless signcryption scheme is based on the Encrypt-then-Sign method discussed in [6]. The scheme presents stronger security properties than one might expect from its internal building-blocks: by sharing randomness between encryption and signature modules not only do we gain extra savings on computational and bandwidth load, but also we obtain strong insider security guarantees. Additionally, we identify a problem in using Coron's technique [36] for tighter security reductions in public-key signature proofs which is specific to the certificateless setting and propose a technique to overcome it.

The certificateless signature scheme proposed in [5], which lacked a security

proof, was found to be vulnerable to key replacement attacks in [55]. Other certificateless signature schemes have been proposed in recent years. We refer to the work of Zhang et al. [90], which proposes a certificateless signature scheme closely based on the identity-based signature scheme of Libert and Quisquater [64]. We base our certificateless signcryption construction on the signature scheme in [90], but it is worthwhile to mention that the proof of security presented for the scheme in [90] is flawed. Despite this, the scheme is secure, but in a slightly weaker security model later proposed informally in [54]. We will discuss this issue further in Section 8.4.

Although the concept of certificateless signcryption has not been previously addressed in literature, a closely related construction, which offers authenticated certificateless encryption functionality was proposed in [35][4]. The difference between authenticated encryption and signcryption is a subtle but significant one: insider attacks are not considered in the unforgeability game, which means that non-repudiation is not guaranteed. The security models adopted in [35] are also significantly weaker than the ones considered in our work, as they do not consider decryption on replaced public keys and impose unreasonable restrictions on the adversary's adaptive behaviour. In particular the forward-secrecy argument presented for the scheme in [35] considers only chosen-plaintext attacks. We base our signcryption scheme on this certificateless encryption scheme, but our Encrypt-then-Sign construction with randomness-reuse permits obtaining improved security guarantees.

[4] Another paper titled "Authenticated Certificateless Public Key Encryption" by Lee and Lee has appeared on IACR ePrint archive in 2004. However, the security models in this paper do not capture the certificateless setting.

Chapter 2

Preliminaries

2.1 Basic Notation and Concepts

In this section we explain some of the fundamental notions and notations that will be used throughout this work.

2.1.1 Algorithmic Notation

If S is a set, we write $v \leftarrow S$ for the action of sampling from the uniform distribution on S and assigning the result to the variable v. We write $v_1, \ldots, v_n \leftarrow S$ for the independent assignments $v_1 \leftarrow S; \ldots; v_n \leftarrow S$. If S contains one element s we use $v \leftarrow s$ as shorthand for $v \leftarrow \{s\}$.

We shall be concerned with probabilistic polynomial-time (PPT) algorithms. A probabilistic machine has, in addition to its input and output tapes, access to an extra randomness tape containing bits chosen (uniformly) at random. If A is a PPT algorithm, we denote by $v \leftarrow A(I)$ the action of running A on input I, which is chosen according to a (usually uniform) distribution and assigning the resulting output to the variable v. We use bold-faced variables for vectors and denote the i-th component of a vector $\mathbf{c}$ with $[\mathbf{c}]_i$.

2.1.2 Negligible Functions

Definition 2.1. *A function $f : \mathbb{N} \to \mathbb{R}$ is called negligible if for every positive polynomial P there exists a κ_0 such that for all $\kappa > \kappa_0$ we have*

$$f(\kappa) < \frac{1}{P(\kappa)}.$$

The standard method to reduce the error probability of an imperfect algorithm is to repeat independent execution of it a sufficiently large (but still polynomial) number of times and then return the majority answer as the final output. If the success probability of an algorithm is greater than $1/P(\kappa)$ for all sufficiently large values of κ, then it follows from the Chernoff bound along the lines in [85, Section 7.2] that this procedure will yield a correct answer with a probability close to 1.

2.1.3 Security Parameter

As in the theory of computational complexity, it is meaningless to talk about the difficulty of solving *one* particular task: the algorithm which simply "prints out the correct answer", is a (very fast) algorithm which solves the problem without even looking at its input! One therefore parameterises the set of problem instances and studies the asymptotic behaviour of a potential algorithm. The parameterisation variable, written κ, is known as the *security parameter* in cryptography. This parameter is often the length of the problem instance written in the binary notation. However, when one is setting up a system, κ is passed to setup algorithms in the unary notation, written 1^κ, to enforce a polynomial run-time.

2.1.4 Game Hopping

We will often need to express how a set of algorithms interact with each other. For instance, we will need to express precisely the correctness condition of an en-

cryption scheme: that decryption undoes encryption. We shall adapt the *game* or *experiment* notion for this purpose. A game is simply a set of rules (usually written in pseudo-code) which describes the input/output behaviour of a number of interactive Turing machines. A game could take an input (usually the security parameter) and returns an output (usually a bit indicating success or failure). The desired property is then expressed by requiring the output to satisfy a certain condition.

When defining security, we will be concerned with an *arbitrary* interactive Turing machine, known as the adversary, taking part in the security game. This adversary has usually only *oracle access* to other algorithms in the game: it does not get the code of the other machines (which might include a secret key) and interacts with them by providing them an input and receiving an output back in unit time. We indicate an algorithm A having access to a set of oracles O by A^O. If this set contains a single oracle, we will refer to the oracle itself by O. To quantify the success of an adversary, we define its *advantage* $\mathrm{Adv}(A)$, as (a function of) the probability that the output of the game satisfies a certain condition. The probability is taken over all the random choices made in the game including those of A and its oracles.

As an example, let us express the fundamental notion of indistinguishability (IND) of two distributions in the game notation.

Definition 2.2. *Two distributions X_0 and X_1, with sampling algorithms S_0 and S_1, are called indistinguishable if the advantage of any probabilistic polynomial-time algorithm A is negligible in the following game:*

$IND(\kappa)$

1. $x_0 \leftarrow S_0(1^\kappa)$
2. $x_1 \leftarrow S_1(1^\kappa)$
3. $b \leftarrow \{0,1\}$
4. $b' \leftarrow A(x_b)$

$$\mathrm{Adv}^{\mathtt{IND}}_{X_0,X_1}(A)(\kappa) := |2\Pr[b = b'] - 1|.$$

We usually omitted the input to the game, κ, for the ease of notation. Note also that the above advantage is a function of κ and could be recast, after dropping κ, as:

$$\mathrm{Adv}^{\mathtt{IND}}_{X_0,X_1}(A) = |\Pr[b' = 1|b = 1] - \Pr[b' = 1|b = 0]|,$$

measuring the "bias" of the distinguisher A.

Sometimes it is difficult to work with a particular game. In such cases one will need to "deform" the game into another game such that the behaviour of any algorithm taking part in the new game is negligibly different from that in the original game. This method, referred to as game hopping, is discussed further in a tutorial by Shoup [87] and in another by Bellare and Rogaway [18].

One particularly useful, and easily proved, method for making a successful hop is through the following observation.

Lemma 2.1 (Difference Lemma)**.** *Let U_1, U_2 and F be events defined on some probability space. Suppose that* $\Pr[U_1 \wedge \neg F] = \Pr[U_2 \wedge \neg F]$*, then*

$$|\Pr[U_1] - \Pr[U_2]| \leq \Pr[F].$$

If U_1 and U_2 denote the events that an adversary is successful in games $\mathtt{Game}_1$ and $\mathtt{Game}_2$ respectively, and if the probability that event F is triggered is negligible,

then one can safely substitute $\texttt{Game}_2$ for $\texttt{Game}_1$. After a series of such game hops, one arrives at a game which is relatively easy to deal with. If the advantage of any algorithm can be shown to be negligible in this final game, and if all the hops made were also negligible, it would follow that the advantage of an arbitrary algorithm is also negligible in the original game.

2.1.5 Computational Group Schemes

To specify a "working group", one needs to define the algorithms for performing computation with the group elements. These specifications, which could be somewhat technical in nature, are referred to as a *computational group scheme*. We only informally mention that such a scheme, denoted by $\Gamma[\hat{G}, G, P, p]$, where $\hat{G}$ is a finite Abelian group and G is a subgroup of prime order p with generator P in $\hat{G}$, specifies a sequence of group distributions for every value of the security parameter κ. This scheme has descriptions of several algorithms for computing and testing encoding of group elements, performing group operations, testing subgroup membership and statistically-close sampling from the groups. The encoding of group elements is assumed to be unique. We refer the interested reader to [37] for the details. The group schemes that we will be working with will be clear from context and hence we shall tacitly compress the above technicalities into "let Γ be a computational group scheme" and use Γ and G interchangeably throughout this work. Group operations, except those in finite fields, are written additively.

2.2 Intractability Assumptions

The security of cryptosystems that we study in this thesis are based on certain problems which are assumed to be computationally hard to solve. In the coming sections we will define these assumptions precisely.

2.2.1 Discrete Logarithm and Related Problems

The fundamental hard problem underlying all intractability assumptions in this work is the discrete logarithm problem.

Definition 2.3 (Discrete Logarithm (DL))**.** *A computational group scheme* Γ *(or as discussed above, simply a group G) satisfies the discrete logarithm assumption if all PPT algorithms A have negligible advantage in the following game.*

DL

1. $a \leftarrow \mathbb{Z}_p^*$
2. $a' \leftarrow A(\Gamma, P, a \cdot P)$

$$\mathrm{Adv}_{\Gamma}^{\mathtt{DL}}(A) := \Pr[a = a'].$$

Although P is implicit in Γ, we pass it explicitly to A to emphasise that discrete logarithms should be computed with respect to this generator.

Definition 2.4 (Computational Diffie–Hellman (CDH))**.** *A computational group scheme* Γ *satisfies the computational Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

CDH

1. $a, b \leftarrow \mathbb{Z}_p^*$
2. $Q \leftarrow A(\Gamma, P, a \cdot P, b \cdot P)$

$$\mathrm{Adv}_{\Gamma}^{\mathtt{CDH}}(A) := \Pr[Q = ab \cdot P].$$

Definition 2.5 (Decisional Diffie–Hellman (DDH))**.** *A computational group scheme* Γ *satisfies the decisional Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

DDH

1. $a, b, c \leftarrow \mathbb{Z}_p^*$
2. $Q_0 \leftarrow cP$; $Q_1 \leftarrow abP$
3. $\beta \leftarrow \{0,1\}$
4. $\beta' \leftarrow A(\Gamma, P, a \cdot P, b \cdot P, Q_\beta)$

$$\mathrm{Adv}_\Gamma^{\mathtt{DDH}}(A) := |2\Pr[\beta' = \beta] - 1|.$$

Okamoto and Pointcheval [71] introduced a new class of intractability assumptions referred to as "gap" problems. Very informally, the gap version of a problem is the problem of solving the computational version with the help of an oracle which solves the decisional version.

Definition 2.6 (Gap Diffie–Hellman (GDH))**.** *A computational group scheme* Γ *satisfies the gap Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

GDH

1. $a, b \leftarrow \mathbb{Z}_p^*$
2. $Q \leftarrow A^O(\Gamma, P, a \cdot P, b \cdot P)$

$$\mathrm{Adv}_\Gamma^{\mathtt{GDH}}(A, q_{\mathtt{DDH}}) := \Pr[Q = ab \cdot P].$$

Here O *is an oracle that, given a tuple* (Γ, xP, yP, Q) *returns* 1 *if* $Q = xyP$ *and* 0 *otherwise. The quantity* $q_{\mathtt{DDH}}$ *denotes the maximum number of queries that A can place to the oracle.*

If we denote by "$\leq_P$" the complexity-theoretic notion of (Cook) reduction, we have DDH $\leq_P$ CDH $\leq_P$ DL, or alternatively.

$$\mathrm{Adv}_\Gamma^{\mathtt{DL}} \leq \mathrm{Adv}_\Gamma^{\mathtt{CDH}} \leq \mathrm{Adv}_\Gamma^{\mathtt{DDH}}.$$

where $\mathrm{Adv}^{*}_{\Gamma}$ denotes the maximum of advantages over all PPT algorithms for problem $*$.

2.2.2 Bilinear Groups

Some of our constructions will use bilinear maps and bilinear groups. We briefly review the necessary facts about these. Further details may be found in [23, 25].

Let G_1, G_2 and G_T be groups with group scheme $\Gamma[G_1, G_2, G_T, e, \rho, p, P_1, P_2]$ satisfying the following properties.

- G_1 and G_2 are, additively written, groups of prime order p, with generators P_1 and P_2. They have efficiently computable group operations.
- There is an efficiently computable isomorphism $\rho : G_2 \to G_1$, with $\rho(P_2) = P_1$.
- There is an efficiently computable bilinear map $e : G_1 \times G_2 \to G_T$, where G_T is a multiplicatively written group of order p.

We have stipulated that our groups should have a bilinear map, $e : G_1 \times G_2 \to G_T$, this is a map which satisfies the following conditions:

1. Bilinear: Given any $Q \in G_1$, $W \in G_2$ and $a, b \in \mathbb{Z}_p$ we have

$$e(aQ, bW) = e(Q, W)^{ab}.$$

2. Non-degenerate: $e(P_1, P_2) \neq 1$.

In many cases one can set $G_1 = G_2$ as is done in [23]. When this is so, we can take ρ to be the identity map; however, to take advantage of certain families of groups [69], we do not restrict ourselves to this case. The map e is usually derived from the Weil or Tate pairings on an elliptic curve.

Definition 2.7 (Computational Bilinear Group Scheme)**.** *If the above conditions hold, we say that* Γ *is a computational bilinear group scheme.*

There are several hard problems associated with bilinear groups which we are interested in for building cryptosystems. These have their origins in the work of Boneh and Franklin [23].

Definition 2.8 (Computational Bilinear Diffie–Hellman (CBDH))**.** *A computational bilinear group scheme* Γ *satisfies the computational bilinear Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

CBDH

1. $a, b, c \leftarrow \mathbb{Z}_p^*$

2. $t \leftarrow A(\Gamma, aP_2, bP_2, cP_2)$

$\mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(A) := \Pr[t = e(P_1, P_2)^{abc}]$.

There are various formulations of the above problem. For example if the input instance is in $G_1 \times G_2 \times G_1$, one can reduce it to the above problem via the isomorphism ρ. Note, however, that an equivalence result would require the map ρ^{-1} to be efficiently computable.

Definition 2.9 (Decisional Bilinear Diffie–Hellman Problem (DBDH))**.** *A computational bilinear group scheme* Γ *satisfies the decisional bilinear Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

DBDH

1. $a, b, c, d \leftarrow \mathbb{Z}_p^*$
2. $Q_0 \leftarrow e(P_1, P_2)^d$; $Q_1 \leftarrow e(P_1, P_2)^{abc}$
3. $\beta \leftarrow \{0, 1\}$
4. $\beta' \leftarrow A(\Gamma, aP_1, bP_2, cP_1, Q_\beta)$

$$\mathrm{Adv}_\Gamma^{\mathtt{DBDH}}(A) := |2\Pr[\beta' = \beta] - 1|.$$

Definition 2.10 (Gap Bilinear Diffie–Hellman (GBDH))**.** *A computational bilinear group scheme Γ satisfies the gap bilinear Diffie–Hellman assumption if all PPT algorithms A have negligible advantage in the following game.*

GBDH

1. $a, b, c \leftarrow \mathbb{Z}_p^*$
2. $t \leftarrow A^O(\Gamma, aP_2, bP_2, cP_2)$

$$\mathrm{Adv}_\Gamma^{\mathtt{GBDH}}(A, q_{\mathtt{DBDH}}) := \Pr[t = e(P_1, P_2)^{abc}].$$

Here O is an oracle that given a tuple $(\Gamma, xP_1, yP_2, zP_1, s)$ returns 1 if $s = e(P_1, P_2)^{xyz}$ and 0 otherwise. The quantity $q_{\mathtt{DBDH}}$ denotes the maximum number of queries that A can place to the oracle.

Definition 2.11 (Modified Gap Diffie–Hellman (mGDH))**.** *Given a computational bilinear group scheme Γ, we say the computational Diffie–Hellman assumption in the presence of a decision bilinear Diffie–Hellman oracle holds in G_1 if all PPT algorithms A have negligible advantage in the following game.*

mGDH

1. $a, b \leftarrow \mathbb{Z}_p^*$
2. $Q \leftarrow A^O(\Gamma, aP_1, bP_1)$

$$\mathrm{Adv}_\Gamma^{\mathtt{mGDH}}(A, q_{\mathtt{DBDH}}) := \Pr[Q = ab \cdot P].$$

Here O and $q_{\mathtt{DBDH}}$ are as in Definition 2.10.

And finally:

Definition 2.12 (q-Bilinear Diffie–Hellman Inversion (q-BDHI))**.** *A computational bilinear group scheme Γ satisfies the q-bilinear Diffie–Hellman inversion assumption if all PPT algorithms A have negligible advantage in the following game.*

q-BDHI

1. $a \leftarrow \mathbb{Z}_p^*$
2. $t \leftarrow A(\Gamma, aP_2, a^2P_2, \ldots, a^qP_2)$

$$\mathrm{Adv}_{\Gamma}^{q-\mathtt{BDHI}}(A) := \Pr[t = e(P_1, P_2)^{1/a}].$$

We have the following reductions between the above problems: DBDH, GBDH $\leq_P$ CBDH and q-BDHI $\leq_P$ $(q-1)$-BDHI $\leq_P \cdots \leq_P$ 1-BDHI, and the latter could be shown to be polynomially equivalent to the CBDH problem.

2.3 Cryptographic Primitives

In this section we introduce some of the basic cryptographic primitives which will be used throughout this thesis.

2.3.1 Hash Functions and the Random Oracle Methodology

Hash functions are ubiquitous objects in cryptography. Informally, a hash function is an efficiently computable, "randomly behaving" function H from a domain A, which is usually the space of arbitrary bit strings $\{0,1\}^*$, to a co-domain B which could be $\{0,1\}^n$, the space of bit strings of length n, or an algebraic structure such as a group.

This random behaviour could be stated in a precise way through various security notions such as one-wayness, collision resistance, entropy smoothing and

so forth. However, these notions have not been very successful in reducing the security of many efficient and practical schemes (which have resisted attacks) to standard complexity-theoretic assumptions.

Bellare and Rogaway [17] proposed a model for hash functions known as the random oracle model (ROM). This model enables one to prove the security of many cryptographic constructions by treating hash functions as *ideal*. An ideal hash function is a function chosen uniformly at random from the set of all functions mapping from A to B. Since such a function would almost surely have an exponential description, it cannot be implemented in practice. Hence the random oracle model is a non-standard model of computation. However, given that intuitively many hash functions are expected to behave randomly, this model gives good heuristic evidence for the security of many schemes.

Random oracles are used in the following way in security proofs. All parties are given oracle access to a random oracle. The protocol is then shown secure with respect to this oracle by using the following (extra) properties of the random oracle:

- Extractability: The input/output values to the random oracle are observable: the adversary has indirect access to the oracle.
- Programmability: The output values of the random oracle could be chosen in a particular way, as long as they maintain a close enough distribution to that of a random function.

One then replaces the random oracle with a concrete hash function.

Note that in the standard model of computation the adversary has access to the code of hash functions whereas this piece of information is taken away from the adversary in the ROM. In fact, this property was used by Canetti et al. [28] to identify conceptual difficulties associated with the ROM: there are (contrived) schemes which are secure in the ROM but are insecure when the random oracles

are instantiated with *any* concrete hash function. A random oracle model proof should therefore be considered as a sanity check or a method to ensure that there are no "embarrassing attacks" against the scheme.

We refer to the model where uninstantiatable mathematical objects such as random oracles are not used as the *standard model*.

2.3.2 Public-Key Encryption Schemes

A public-key encryption (PKE) scheme [15] is defined formally by a triple of PPT algorithms $(\mathbb{G}_{\mathtt{PKE}}, \mathbb{E}_{\mathtt{PKE}}, \mathbb{D}_{\mathtt{PKE}})$ as follows.

- $\mathbb{G}_{\mathtt{PKE}}(1^{\kappa})$: This is the probabilistic key generation algorithm which takes as input the security parameter 1^{κ} and returns a secret key $\mathtt{SK}$ and a matching public key $\mathtt{PK}$.

- $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r)$: This is the probabilistic encryption algorithm, which on input a message $\mathtt{m} \in \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$, a public key $\mathtt{PK}$ and possibly some randomness $r \in \mathcal{R}_{\mathtt{PKE}}(\mathtt{PK})$ outputs a ciphertext $\mathtt{c} \in \mathcal{C}_{\mathtt{PKE}}(\mathtt{PK})$.

- $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$: This is the deterministic decryption algorithm. On input a ciphertext $\mathtt{c}$ and a private key $\mathtt{SK}$ this algorithm outputs a message $\mathtt{m} \in \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$ or a special failure symbol $\perp \notin \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$.

Let us clarify a few points regarding the above definition, which also apply to other primitive definitions later on.

- We do not explicitly define setup algorithms which define the domain parameters for the scheme such as the underlying groups. This is simply to reduce the amount of notation. Our results can be expanded to cope with this by inserting a domain parameter generation algorithm `Setup` which takes as input 1^{κ} and outputs a global or domain parameter I which is then passed to

the key generation algorithm. We assume that I as well as the security parameter 1^κ are included in public keys. Various spaces such as the message, randomness and ciphertext spaces are parameterised by public keys and are denoted by $\mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$, $\mathcal{R}_{\mathtt{PKE}}(\mathtt{PK})$ and $\mathcal{C}_{\mathtt{PKE}}(\mathtt{PK})$ respectively. These spaces could vary from one public key to another, as is the case with the RSA encryption scheme, or could only depend on the domain parameter I, as in the ElGamal encryption scheme.

- When the randomness r is not explicitly passed to an algorithm, we assume it is chosen uniformly at random from appropriate randomness space.

- A decryption algorithm could be probabilistic. Kiltz [58] describes a randomised decryption algorithm for one of his schemes which performs better than the deterministic alternative. Since we will not be dealing with randomised decryption algorithms, we shall assume that they are deterministic throughout this thesis.

We capture the fact that decryption undoes encryption formally by requiring that the following experiment returns 1 with probability 1.

1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^\kappa)$
2. $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$
3. $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK})$
4. $\mathtt{m}' \leftarrow \mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$
5. If $\mathtt{m}' = \mathtt{m}$ return 1 else return 0

If this is the case, the scheme is called *correct*. Observe that we have required perfect correctness: any ciphertext produced by the genuine encryption algorithm should always decrypt. All our concrete constructions do indeed satisfy this condition, but our general results can be extended to cover schemes with weaker cor-

rectness definitions, which allows a negligible failure in decryption, in the standard way [37]. Correctness condition for many other cryptographic primitives that appear in this work will be clear from the context, and we will omit an explicit game-based definition for the ease of exposition.

To cope with probabilistic encryption schemes, we sometimes require that not too many choices for r encrypt a given message to a given ciphertext. Let $\gamma(\mathtt{PK})$ be the least upper bound such that

$$|\{r \in \mathcal{R}_{\mathtt{PKE}}(\mathtt{PK}) : \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r) = c\}| \leq \gamma(\mathtt{PK})$$

for every $\mathtt{m} \in \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$ and $\mathtt{c} \in \mathcal{C}_{\mathtt{PKE}}(\mathtt{PK})$. Our requirement is that the quantity $\gamma(\mathtt{PK})/|\mathcal{R}_{\mathtt{PKE}}(\mathtt{PK})|$ is a negligible function of the security parameter. This condition is sometimes referred to as γ-*uniformity*.

Let us now describe the various notions of security of a PKE scheme. Consider the following two-stage games between an adversary $A = (A_1, A_2)$ of the encryption algorithm and a hypothetical challenger.

OW-atk	IND-atk
1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^\kappa)$	1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(1^\kappa)$
2. $\mathrm{st} \leftarrow A_1^{O_1}(\mathtt{PK})$	2. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{PK})$
3. $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$	3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK})$	4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_b, \mathtt{PK})$
5. $\mathtt{m}' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$	5. $b' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$
$\mathrm{Adv}^{\mathtt{OW-atk}}_{\mathtt{PKE}}(A) := \Pr[\mathtt{m}' = \mathtt{m}]$.	$\mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{PKE}}(A) := \lvert 2\Pr[b' = b] - 1\rvert$.

In the above games st is some state information and we require $\mathtt{m}_0$ and $\mathtt{m}_1$ to be of equal length[1]. O_i denotes the (set of) oracles to which the adversary has access.

[1]This condition on message lengths and the fact that st denotes some state information apply to all games. We shall henceforth omit it.

There are various possibilities for this oracle depending on the attack model $\mathtt{atk} \in \{\mathtt{CPA}, \mathtt{CPA}^{++}, \mathtt{CCA1}, \mathtt{CCA}\}$:

- $\mathtt{atk} = \mathtt{CPA}$: In this model, which captures chosen plaintext attacks, the oracles are all null[2].
- $\mathtt{atk} = \mathtt{CPA}^{++}$: In this model the adversary is given access to the following oracles:
  - A ciphertext validity oracle, which on input a ciphertext $\mathtt{c}$ returns 0 if $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK}) = \perp$ and 1 otherwise. Public-key encryption schemes for which an explicit algorithm exists to implement a plaintext/ciphertext checking oracle will be called *verifiable*.
  - A plaintext/ciphertext checking oracle, which on input a message $\mathtt{m}$ and a ciphertext $\mathtt{c}$ returns 1 if $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK}) = \mathtt{m}$ and 0 otherwise.
- $\mathtt{atk} = \mathtt{cCPA}^{++}$: In this model, referred to as the complete CPA^{++} model, the adversary is given access to the following oracles:
  - A complete ciphertext validity oracle, which on input $(\mathtt{c}, \mathtt{PK})$ returns 1 if $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK}) \neq \perp$ and 0 otherwise. Here $\mathtt{SK}$ in the secret key corresponding to the public key $\mathtt{PK}$. We require the challenger to *find* $\mathtt{SK}$ if necessary. This oracle returns 0 if $\mathtt{PK}$ is not a valid public key. Public-key encryption schemes for which an explicit algorithm exists to implement this oracle will be called *completely verifiable*.
  - A complete plaintext/ciphertext checking oracle, which takes $(\mathtt{c}, \mathtt{PK}, \mathtt{m})$ and returns 1 if $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK}) = \mathtt{m}$ and 0 otherwise. The points made above also apply here.

[2] A null oracle outputs the empty string on any input.

- $\mathtt{atk} = \mathtt{CCA1}$: In this model, which captures non-adaptive chosen ciphertext attacks, the oracle O_1 is a decryption oracle with respect to the public key $\mathtt{PK}$ and O_2 is the null oracle.

- $\mathtt{atk} = \mathtt{CCA}$: In this model, which captures adaptive chosen ciphertext attacks, the oracles O_1 and O_2 are decryption oracles with respect to the public key $\mathtt{PK}$, subject to the restriction that O_2 cannot be queried on $\mathtt{c}^*$.

Note that the goal of the adversary in the game on the left is to fully recover the message underlying the challenge ciphertext whereas its goal in the other game is easier: it only needs to distinguish which of the two messages is encrypted inside $\mathtt{c}^*$.

A public-key encryption algorithm is considered to be secure, in the sense of a given goal and attack model (IND-CCA for example) if, for all PPT adversaries, the advantage in the relevant game above is a negligible function of the security parameter[3].

In the original work [20], a ciphertext equality oracle, which on input $(\mathtt{c}_1, \mathtt{c}_2)$ returns 1 if $\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}_1, \mathtt{SK}) = \mathbb{D}_{\mathtt{PKE}}(\mathtt{c}_2, \mathtt{SK})$ and 0 otherwise, is also required to exist. The proof which appears in Section 7.2.2 is a corrected version where this oracle is no longer needed. We call the model with this extra oracle the CCA$^-$ model.

Dent [38] calls the attack model CPA^{++} in which an adversary has access only to the ciphertext validity checking oracle the CPA$^+$ model (which motivates our naming). The CPA$^+$ model was first used in [56] to attack a version of the EPOC-2 public-key encryption scheme; it is sometimes referred to as a "reaction attack".

As we shall see in Chapter 7, one of our generic constructions only requires a scheme which is OW-CPA^{++}. Such schemes are readily available, for example, the naive text-book RSA scheme is such a scheme, assuming the RSA problem is

[3]Once a security game and its advantage are formulated, security is defined by this requirement. We shall henceforth omit it to avoid repetition.

hard. As another example, one can take text-book ElGamal, on the assumption that the gap Diffie–Hellman problem is hard. Textbook RSA is also completely verifiable, as is textbook ElGamal, if one implements it, for instance, on a group which admits a bilinear map. Note that in the latter case the DDH problem can be solved using the bilinear map and hence the security of the scheme cannot rest on this assumption.

The CPA^{++} attack model will only be used (in Chapter 7) in the one-way context and hence we do not need to restrict the above oracles. One can formulate CPA^{++} and/or CCA^{-} models in the IND setting. One can then ask if there exist schemes which are IND-CCA^{-} but not IND-CCA secure.

Note, the way the oracles in CPA^{++} and CCA^{-} models are defined. The ciphertext validity oracle, for instance, is not defined by requiring it to return 1 if there is a message $\mathtt{m}$ and random coins r such that $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r) = \mathtt{c}$. As we shall see, these oracles will be used to simulate a consistent decryption oracle and hence we have adapted the formulation above.

Shoup [86] and An et al. [6] consider another security notion called the generalised CCA model (gCCA). In this model, the restriction on the decryption oracle in the second stage of the game is extended to all ciphertexts which decrypt to the same message as that implicit in the challenge ciphertext. This definition allows the so-called *benignly malleable* schemes to be classified as secure. An et al. [6] argue that the gCCA model is a more natural definition of security compared to the CCA model. The ECIES scheme is only required to be IND-gCCA secure in some standards [86].

2.3.3 Key Encapsulation Mechanisms

A standard KEM [37] is defined by a triple of PPT algorithms.

- $\mathbb{G}_{\mathtt{KEM}}(1^{\kappa})$: This is the probabilistic key generation algorithm. This takes as

input 1^κ and outputs a secret/public key pair $(\mathtt{SK}, \mathtt{PK})$ as well as the descriptions of key space $\mathcal{K}_{\mathtt{KEM}}(\mathtt{PK})$ and encapsulation space $\mathcal{C}_{\mathtt{KEM}}(\mathtt{PK})$.

- $\mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$: This is the probabilistic key encapsulation algorithm. This takes as input $\mathtt{PK}$ and outputs $(\mathtt{k}, \mathtt{c}) \in \mathcal{K}_{\mathtt{KEM}}(\mathtt{PK}) \times \mathcal{C}_{\mathtt{KEM}}(\mathtt{PK})$. The item $\mathtt{c}$ is called the encapsulation of the (session) key $\mathtt{k}$. The key $\mathtt{k}$ is assumed to be uniformly distributed over the key space $\mathcal{K}_{\mathtt{KEM}}(\mathtt{PK})$.

- $\mathbb{D}_{\mathtt{KEM}}(\mathtt{c}, \mathtt{SK})$: This is the deterministic decapsulation algorithm. On input a user secret key $\mathtt{SK}$ and an encapsulation $\mathtt{c}$ this outputs the corresponding value of $\mathtt{k}$ or an invalid encapsulation symbol $\perp$.

The security of a KEM is defined according to the following two-stage games[4]:

OW-atk

1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{KEM}}(1^\kappa)$
2. $\mathrm{st} \leftarrow A_1^{O_1}(\mathtt{PK})$
3. $(\mathtt{k}, \mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$
4. $\mathtt{k}' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}^{\mathtt{OW-atk}}_{\mathtt{KEM}}(A) := \Pr[\mathtt{k}' = \mathtt{k}].$

IND-atk

1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{KEM}}(1^\kappa)$
2. $\mathrm{st} \leftarrow A_1^{O_1}(\mathtt{PK})$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{KEM}}(\mathtt{PK})$
4. $(\mathtt{k}_1, \mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b, \mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{KEM}}(A) := |2\Pr[b' = b] - 1|.$

Here the oracles O_i denote a decapsulation oracle with respect to the public key $\mathtt{PK}$. In the CPA attack model the adversary is not allowed any access to O_i, while in the CCA attack model it does have access to O_i, subject to the restriction that in the second phase A is not allowed to call O_2 on the challenge encapsulation $\mathtt{c}^*$.

[4]It is shown in [70] that non-malleability under chosen ciphertext attacks is equivalent to IND-CCA for KEMs.

The following alternative security model for KEMs, which is in some sense more natural as it provides both keys for the adversary, is shown to be equivalent to the IND-atk definition in Lemma 2.2.

IND-atk′

1. $(\mathtt{PK}, \mathtt{SK}) \leftarrow \mathbb{G}_{\mathtt{KEM}}(1^{\kappa})$
2. $\mathrm{st} \leftarrow A_1^{\mathcal{O}_1}(\mathtt{PK})$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{KEM}}(\mathtt{PK})$
4. $(\mathtt{k}_1, \mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathtt{k}_b, \mathtt{k}_{1-b}, \mathrm{st})$

The advantage is defined as before.

We remark that the above security models can be simplified by omitting the first phase. The resulting game is unchanged in the CPA case and is statistically-close to the original definition when in the CCA attack model. This follows from the fact that there is a negligible chance that the adversary could query the challenge ciphertext to the decryption oracle in the first stage of the game.

Let us prove the equivalence of the atk and atk′ alternatives of KEM security games. This will also serve as a warm-up to the reductions which will be presented in this work.

Lemma 2.2. *The IND-atk and IND-atk′ security models are equivalent.*

Proof. Let us first prove IND-atk′ security implies IND-atk security. Let A be an adversary against the IND-atk game. We construct an adversary B against the IND-atk′ game as follows. Algorithm B on receiving a public key $\mathtt{PK}$ passes it onto A answering any decapsulation oracle queries using its own equivalent oracle. When A outputs st, algorithm B also outputs this to receive a challenge $(\mathtt{c}^*, \mathtt{k}_b, \mathtt{k}_{1-b}, \mathrm{st})$. Al-

gorithm B now discards the third component $\mathtt{k}_{1-b}$ and returns the triple $(\mathtt{k}_b, \mathtt{c}^*, \mathrm{st})$ to A as the challenge. Oracle queries are answered as in the first stage. When B receives a bit b' it outputs this as its guess. Since the challenge bit is unchanged it is clear that:

$$\mathrm{Adv}^{\mathtt{IND-atk'}}_{\mathtt{KEM}}(B) = \mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{KEM}}(A).$$

We now prove that IND-atk security implies IND-atk$'$ security. Let A be an adversary against the IND-atk$'$ game. We construct an adversary B against the IND-atk game as follows. Algorithm B on receiving the public key $\mathtt{PK}$ returns it to A, answering any decapsulation queries using its own equivalent oracle. When A outputs st, algorithm B returns this to its challenge oracle to get $(\mathtt{k}^*, \mathtt{c}^*, \mathrm{st})$. Now B generates a key $\mathtt{k} \leftarrow \mathcal{K}_{\mathtt{KEM}}(\mathtt{PK})$ and a random bit σ. It then sets $\mathtt{k}_0 = \mathtt{k}^*$ and $\mathtt{k}_1 = \mathtt{k}$ and passes $(\mathtt{c}^*, \mathtt{k}_\sigma, \mathtt{k}_{1-\sigma}, \mathrm{st})$ to A, answering any oracle queries as before. When A returns a bit b', algorithm B returns 1 if $\sigma = b'$ and 0 otherwise.

Now when $b = 1$, A is ran in the IND-atk$'$ game environment, with its challenge bit being σ. Hence,

$$\mathrm{Adv}^{\mathtt{IND-atk'}}_{\mathtt{KEM}}(A) = |\Pr[b' = 1 | \sigma = 1 \wedge b = 1] - \Pr[b' = 1 | \sigma = 0 \wedge b = 1]|.$$

Also

$$\begin{aligned} 2\Pr[b' = \sigma | b = 1] &= \Pr[b' = 1 | \sigma = 1 \wedge b = 1] + \Pr[b' = 0 | \sigma = 0 \wedge b = 1] \\ &= \Pr[b' = 1 | \sigma = 1 \wedge b = 1] + 1 - \Pr[b' = 1 | \sigma = 0 \wedge b = 1]. \end{aligned}$$

Therefore

$$\mathrm{Adv}^{\mathtt{IND-atk'}}_{\mathtt{KEM}}(A) = |2\Pr[b' = \sigma | b = 1] - 1|.$$

Also $\Pr[b' = \sigma | b = 0] = \frac{1}{2}$ by information theoretic considerations (σ is indepen-

dent of the adversary's view). We obtain the result by observing:

$$\begin{aligned} 2\mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{KEM}}(B) &= |2\Pr[b' = \sigma | b = 1] - 2\Pr[b' = \sigma | b = 0]| \\ &= |2\Pr[b' = \sigma | b = 1] - 1| \\ &= \mathrm{Adv}^{\mathtt{IND-atk'}}_{\mathtt{KEM}}(A). \end{aligned}$$

□

2.3.4 Data Encapsulation Mechanisms

A secret-key encryption (SKE) scheme [57] is specified by three PPT algorithms as follows.

- $\mathbb{G}_{\mathtt{SKE}}(1^\kappa)$: This is the probabilistic key generation algorithm which on input a security parameter 1^κ outputs a key $\mathtt{k} \in \mathcal{K}_{\mathtt{SKE}}(\kappa)$. As usual, the key, message and ciphertext spaces are given by $\mathcal{K}_{\mathtt{SKE}}(\kappa)$, $\mathcal{M}_{\mathtt{SKE}}(\kappa)$ and $\mathcal{C}_{\mathtt{SKE}}(\kappa)$ respectively.

- $\mathbb{E}_{\mathtt{SKE}}(\mathtt{m},\mathtt{k})$: This is the probabilistic encapsulation algorithm which on input a message $\mathtt{m} \in \mathcal{M}_{\mathtt{SKE}}(\kappa)$ and a key $\mathtt{k} \in \mathcal{K}_{\mathtt{SKE}}(\kappa)$, outputs a ciphertext $\mathtt{c}$.

- $\mathbb{D}_{\mathtt{SKE}}(\mathtt{c},\mathtt{k})$: This is the deterministic decryption algorithm which on input a ciphertext $\mathtt{c}$ and a key $\mathtt{k} \in \mathcal{K}_{\mathtt{SKE}}(\kappa)$ outputs a message $\mathtt{m} \in \mathcal{M}_{\mathtt{SKE}}(\kappa)$ or a failure symbol $\perp$.

An SKE scheme is called *correct* if $\mathbb{D}_{\mathtt{SKE}}(\mathbb{E}_{\mathtt{SKE}}(\mathtt{m},\mathtt{k}),\mathtt{k}) = \mathtt{m}$ for all keys $\mathtt{k}$ and messages $\mathtt{m}$.

The security of an SKE scheme is defined through games similar to those in the public-key setting.

IND-atk

1. $\mathtt{k} \leftarrow \mathbb{G}_{\mathtt{SKE}}(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1^{O_1}(1^\kappa)$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{SKE}}(\mathtt{m}_b, \mathtt{k})$
5. $b' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{SKE}}(A) := |2\Pr[b' = b] - 1|.$

FG-atk

1. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1(1^\kappa)$
2. $\mathtt{k} \leftarrow \mathbb{G}_{\mathtt{DEM}}(1^\kappa)$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{DEM}}(\mathtt{m}_b, \mathtt{k})$
5. $b' \leftarrow A_2^{O}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}^{\mathtt{FG-atk}}_{\mathtt{DEM}}(A) := |2\Pr[b' = b] - 1|.$

The oracles O_i in the IND-atk game on the left denote an encryption and, in the CCA setting, a decryption oracle subject to the usual restriction that O_2 cannot be queried on $\mathtt{c}^*$. In contrast to the public-key setting, the encryption oracle is necessary to enable the adversary to perform encryption on plaintexts of its choice.

The IND-atk security notion is stronger than what we will need in our work. Note that in the KEM-DEM construction (see Section 2.4.1), the session key is chosen after the message and it is only used once. This observation leads to a relaxed security model, presented above on the right, where only a second stage attack is permitted. In this game, known as the Find-then-Guess (FG) game, O in the CCA setting, is a decryption oracle with respect to the key $\mathtt{k}$. The adversary does not have access to an encryption oracle anymore. A secret-key encryption scheme used in this weaker context is referred to as a *data encapsulation mechanism* (DEM).

As shown in [37], a secure DEM can be constructed using the one-time pad, with its key expanded using a pseudo-random generator, together with a one-time message authentication code (MAC), which ensures the authenticity of the encrypted data.

2.3.5 Identity-Based Encryption Schemes

Here we give the security notions for an identity-based encryption scheme, as first introduced by Boneh and Franklin [23]. An identity-based encryption (IBE) scheme is specified by four PPT algorithms.

- $\mathbb{G}_{\mathtt{IBE}}(1^\kappa)$: A probabilistic algorithm which takes as input the security parameter 1^κ and returns the TA's master secret key $\mathtt{Msk}$ and a matching master public key $\mathtt{Mpk}$[5].

- $\mathbb{X}_{\mathtt{IBE}}(\mathtt{ID},\mathtt{Msk})$: The (possibly) probabilistic private key extraction algorithm which takes as input $\mathtt{Msk}$ and $\mathtt{ID} \in \{0,1\}^*$, an identifier string for a user, and returns the associated private key D.

- $\mathbb{E}_{\mathtt{IBE}}(\mathtt{m},\mathtt{ID},\mathtt{Mpk};r)$: This is the probabilistic encryption algorithm. On input a message $\mathtt{m} \in \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$, an identifier $\mathtt{ID}$, the master public key $\mathtt{Mpk}$ and possibly some randomness $r \in \mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})$, this algorithm outputs a ciphertext $\mathtt{c} \in \mathcal{C}_{\mathtt{IBE}}(\mathtt{Mpk})$.

- $\mathbb{D}_{\mathtt{IBE}}(\mathtt{c},D)$: This is the deterministic decryption algorithm. On input a ciphertext $\mathtt{c}$ and a private key D this outputs a message $\mathtt{m} \in \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$ or a failure symbol $\perp$.

The correctness of an IBE scheme is defined analogously to a PKE scheme by requiring that for all identities decryption undoes encryption given the correct user private key.

Again, to cope with probabilistic IBE schemes, we sometimes require that not too many choices for r encrypt a given message to a given ciphertext. Let $\gamma(\mathtt{Mpk})$ be the least upper bound such that

$$|\{r \in \mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk}) : \mathbb{E}_{\mathtt{IBE}}(\mathtt{m},\mathtt{ID},\mathtt{Mpk};r) = \mathtt{c}\}| \leq \gamma(\mathtt{Mpk})$$

[5]The points on the definition of a PKE scheme in Section 2.3.2 also apply here.

for every $\mathtt{ID} \in \{0,1\}^*$, $\mathtt{m} \in \mathcal{M}_{\mathtt{PKE}}(\mathtt{Mpk})$ and $\mathtt{c} \in \mathcal{C}_{\mathtt{PKE}}(\mathtt{Mpk})$. Our requirement is that the quantity $\gamma(\mathtt{Mpk})/|\mathcal{R}_{\mathtt{PKE}}(\mathtt{Mpk})|$ is a negligible function of the security parameter.

Consider the following two-stage games between an adversary A of the encryption algorithm and a hypothetical challenger.

OW-atk

1. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(1^\kappa)$
2. $(\mathtt{ID}^*, \mathrm{st}) \leftarrow A_1^{\mathcal{O}_1}(\mathtt{Mpk})$
3. $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}^*, \mathtt{Mpk})$
5. $\mathtt{m}' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{OW-atk}}(A) := \Pr[\mathtt{m}' = \mathtt{m}].$

IND-atk

1. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathtt{ID}^*, \mathrm{st}) \leftarrow A_1^{\mathcal{O}_1}(\mathtt{Mpk})$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}_b, \mathtt{ID}^*, \mathtt{Mpk})$
5. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-atk}}(A) := |2\Pr[b' = b] - 1|.$

Here, $\mathcal{O}_i$ are the oracles to which the adversary has access. There are various possibilities for these oracles depending on the attack model:

- In the CPA model (atk=CPA) the adversary only has access to a private key extraction oracle which on input of $\mathtt{ID} \neq \mathtt{ID}^*$ outputs the corresponding value of D.
- In the CCA model (atk=CCA) the adversary has access not only to a private key extraction oracle as above, but also to a decryption oracle with respect to any identity ID of the adversary's choosing. The adversary has access to this decryption oracle, subject to the restriction that in the second phase A is not allowed to call it with the pair $(\mathtt{c}^*, \mathtt{ID}^*)$.

Remark. Although the concrete IBE schemes considered in this thesis have deterministic key extraction algorithms, we mention that one should clearly state the behaviour, on repeated queries, of an oracle corresponding to a probabilistic key

extraction algorithm. For generic results the convention agreed does not usually affect the security argument. For concrete schemes, however, it is usually assumed in the literature that the extraction oracle simply returns the previously computed value [22, 48, 88]. This observation applies to all oracles corresponding to randomised algorithms.

The security model for an IBE scheme could be considered in a weaker setting where the adversary does not choose the target identity $\mathtt{ID}^*$ adaptively. In this *selective* model the attacked identity is chosen at the beginning of the game:

OW-atk	IND-atk
1. $(\mathtt{ID}^*, \mathrm{st}) \leftarrow A_0(1^\kappa)$	1. $(\mathtt{ID}^*, \mathrm{st}) \leftarrow A_0(1^\kappa)$
2. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(1^\kappa)$	2. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(1^\kappa)$
3. $\mathrm{st} \leftarrow A_1^{O_1}(\mathtt{Mpk})$	3. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk}, \mathrm{st})$
4. $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$	4. $b \leftarrow \{0,1\}$
5. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}^*, \mathtt{Mpk})$	5. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}_b, \mathtt{ID}^*, \mathtt{Mpk})$
6. $\mathtt{m}' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$	6. $b' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$

Here $\mathtt{atk} \in \{\mathtt{sCPAs}, \mathtt{sCCA}\}$ and the advantage of an adversary is defined as in the non-selective games. This weaker, but still reasonable, security requirement might allow for a wider range of IBEs to be classified as secure. It is worth noting that a selective-ID secure IBE scheme can often be converted to a fully secure scheme by modelling the hash function applied to identities as a random oracle model [22].

2.3.6 Certificateless Encryption Schemes

We now describe certificateless encryption schemes as first proposed by Al-Riyami and Paterson. We refer the reader to the original works [5, 2, 4] as well as the new work done in this area [41, 39] for further details.

A certificateless scheme makes use of a trusted third party known as a key generation centre (KGC). Unlike the trusted party in an identity-based setting, the KGC does not have access to users' private keys. The KGC uses a global secret key to compute *partial private keys* for users from their identities. Partial private keys are passed from the KGC to the users in a possibly untrusted manner (see [2] for a discussion of the transmission mechanism in more detail). After a user with identity $\mathtt{ID}$ has been supplied with partial private key D by the KGC, it combines D with some additional secret information, its *secret value*, to generate a full private key S. The secret value is not known to the KGC and therefore neither is S. The user can compute its public key $\mathtt{PK}$ without knowing D.

The system is not identity-based: the public key of a user cannot be derived from its identity alone. Instead, a user publishes its public key in some publicly accessible directory. Unlike a traditional public key infrastructure (PKI), it is not necessary to obtain and verify certificates for public keys in this scenario.

Formally, a certificateless scheme is specified by six PPT algorithms.

- $\mathtt{Setup}(1^\kappa)$. This is a global setup algorithm, which takes as input the security parameter 1^κ and returns the Key Generation Centre's (KGC) secret key $\mathtt{Msk}$ and domain parameters I including a master public key $\mathtt{Mpk}$ and descriptions of message space $\mathcal{M}_{\mathtt{CLE}}(I)$, ciphertext space $\mathcal{C}_{\mathtt{CLE}}(I)$ and randomness space $\mathcal{R}_{\mathtt{CLE}}(I)$. This algorithm is executed by the KGC, which publishes I.

- $\mathtt{Extract\text{-}Partial\text{-}Private\text{-}Key}(\mathtt{ID}, \mathtt{Msk}, I)$. A probabilistic algorithm that takes as input $\mathtt{Msk}$, I and an identifier string $\mathtt{ID} \in \{0,1\}^*$ representing a user's identity, and returns a partial private key D. This algorithm is run by the KGC, after verifying the user's identity.

- $\mathtt{Generate\text{-}User\text{-}Keys}(\mathtt{ID}, I)$. A probabilistic algorithm which takes an identity and the domain parameters and outputs a secret value $\mathtt{SK}$ and a public key

PK. This algorithm is run by a user to obtain a public key and a secret value which can be used to construct a full private key. The public key is published without certification.

- Set-Private-Key$(D, \mathtt{SK}, I)$. A deterministic algorithm which takes as input a partial private key D and a secret value SK and returns the full private key S. Again, this algorithm is run by a user to construct the full private decryption key.

- $\mathbb{E}_{\mathtt{CLE}}(\mathtt{m}, \mathtt{PK}, \mathtt{ID}, \mathtt{Mpk}; r)$: This is the probabilistic encryption algorithm. On input a message $\mathtt{m} \in \mathcal{M}_{\mathtt{CLE}}(I)$, a public key PK, an identifier ID, the master public key Mpk, and possibly some randomness $r \in \mathcal{R}_{\mathtt{CLE}}(I)$, this algorithm outputs either a ciphertext $\mathtt{c} \in \mathcal{C}_{\mathtt{CLE}}(I)$ or an error symbol $\perp$.

- $\mathbb{D}_{\mathtt{CLE}}(\mathtt{c}, S)$. This is the deterministic decryption algorithm. On input a ciphertext c and the full private key S this algorithm outputs the corresponding value of the plaintext m or a failure symbol $\perp$.

Owing to the lack of authenticating information for public keys, certificates for example, an adversary may be able to replace users' public keys with public keys of its choice. This appears to give adversaries enormous power; however, to compute the full private key of a user, knowledge of the partial private key is necessary. To capture this scenario, Al-Riyami and Paterson [5] consider a security model in which an adversary is able to adaptively replace users' public keys with public keys of its choice. Such an adversary is called a Type-I adversary.

In a certificateless cryptosystem we also need to ensure the desired escrow-free property of the system holds. An honest-but-curious KGC, who has access to all the user partial private keys, should not be able to eavesdrop on private communication. We refer to an adversary in this setting as a Type-II adversary.

Below we formally describe the two types of adversary discussed above.

IND-atk-x^s

1. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathtt{Setup}(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathtt{ID}^*, \mathrm{st}) \leftarrow A_1^{\mathcal{O}_1}(\mathtt{Mpk}, \mathtt{aux})$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{CLE}}(\mathtt{m}_b, \mathtt{PK}^*, \mathtt{ID}^*, \mathtt{Mpk})$
5. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathrm{st})$

When performing the encryption (step 4) in the games, the challenger uses the *current* public key $\mathtt{PK}^*$ of the user with identifier $\mathtt{ID}^*$. The adversary's advantage is defined to be

$$\mathrm{Adv}_{\mathtt{CLE}}^{\mathtt{IND-atk-x}^s}(A) := |2\Pr[b' = b] - 1|,$$

where x denotes the type of the attack (I or II) and the variable s takes one of the three values $-$, $+$ or the empty string and denotes the strength of the attack. These are referred to as weak, strong and normal Type-x models respectively. In a Type-I game aux is the empty string whereas in Type-II, it is the master secret key Msk.

The oracles to which the adversary has access to are as follows.

- Request public key: on input an ID, return the corresponding public key. This oracle generates a new PK if it does not already exist.
- Replace public key: on input an ID and a valid public key[6] PK, replace the public key of ID with PK.
- Extract partial private key: on input an ID, return a partial private key for that identity.
- Extract full private key: on input an ID, return a full private key for that identity.

[6] In our schemes public keys can be checked for validity.

- Decryption: on input a ciphertext and an identity, this oracle constructs a full private key corresponding to the identity, obtaining a public key for it first, and returns the result of running the decryption algorithm on the ciphertext and the computed full private key. Note that this oracle might be asked to decrypt a ciphertext with respect to a replaced public key. In this case, the decryption oracle is required to first *find* a matching secret value for the public key, and then decrypt the ciphertext using this secret key.

Let us now describe what the oracle restrictions are in each setting.
A strong Type-I adversary should respect the following restrictions:

1. No full private key extraction for $\mathtt{ID}^*$ at any point.
2. No full private key extraction for any identity if the corresponding public key has been replaced.
3. No partial private key extraction for $\mathtt{ID}^*$ if public key of this identity has been replaced before challenge ciphertext was issued.
4. No decryption query on $\mathtt{c}^*$ under $\mathtt{ID}^*$ unless the public key $\mathtt{PK}^*$ used to encrypt $\mathtt{m}_b$ is replaced.

The decryption oracle is restricted further in the weak and normal Type-I settings:

- Normal Type-I: if the adversary has replaced a public key and it subsequently requires a decryption query that involves a decryption with the corresponding secret key, it must supply the associated secret value to the decryption oracle. Note that the decryption oracle continues to use $\mathtt{Msk}$ which is unknown to the adversary.
- Weak Type-I: in this case no decryption queries on replaced public keys are allowed.

The oracle access restrictions that a Type-II adversary should respect are similar to those given in conditions 1–4 above, but with the following extra restrictions:

5. No partial private key extraction for any identity as the adversary can compute these using $\mathtt{aux} = \mathtt{Msk}$.

6. No public key replacement on $\mathtt{ID}^*$ in the first stage.

Normal and weak Type-II adversaries are defined analogously.

Remark. Since the KGC is able to produce partial private keys, it is assumed in the original work [5] that the KGC does not replace users public keys itself. Under this assumption, a user is placing the similar level of trust in a KGC that it would in a PKI certificate authority: it is always assumed that a CA does not issue certificates for individuals on public keys which it has maliciously generated itself! We denote a non-replacing Type-II adversary with Type-II$^{=}$, as it is weaker than a Type-II^{-} adversary. We believe the Type-II model as introduced here is a more natural model. Not only it captures a wider range of attacks by the KGC, it also allows for results such as Lemma 8.1 to be established. Note also that a Type-I adversary in [5] is referred to as a Type-I^{+} adversary here.

There exist stronger models for misbehaving KGCs. The malicious KGC model [7] formalises the setting where the KGC generates the system parameters $(\mathtt{Msk}, \mathtt{Mpk})$ maliciously. A more natural and stronger variant of this model, where the adversary is not required to provide the master secret key, is presented below. The allowed oracles and their restrictions in the Type-M^{s} model are identical to those in the Type-II^{s} model. Note that in order to answer decryption queries in the strong variant the challenger might need to find D as well as $\mathtt{SK}$ [40]. These values should be provided by the adversary in the normal models.

IND-atk-M^s

1. $(\mathtt{Mpk}, \mathrm{st}) \leftarrow A_0(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathtt{ID}^*, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk}, \mathtt{aux}, \mathrm{st})$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathbb{E}_{\mathtt{CLE}}(\mathtt{m}_b, \mathtt{PK}^*, \mathtt{ID}^*, \mathtt{Mpk})$
5. $b' \leftarrow A_2^{O_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}_{\mathtt{CLE}}^{\mathtt{IND-atk-M}^s}(A) := |2\Pr[b' = b] - 1|.$

In each of the above settings, if the adversary does not make any decryption queries, it is said to be an IND-CPA-x adversary. The strength of attack in the CPA models is irrelevant as there are no decryption oracles.

Remark. Whether the strong security models given above are relevant to practice or not remains controversial. The game which requires correct decryption of ciphertexts with replaced public keys cannot be specified in a way which runs in polynomial time: one needs to search through an exponential space to find a correct matching secret value. Given that oracles are intended to model real life procedures which have polynomial run-time, the relevance of these models is somewhat theoretical. The authors in [5] justify these models by stating that they leave a margin of error in the security analysis. Strongly secure CLE schemes are shown to be realisable in the standard model [41]: although the original game does not run in polynomial time, a suitable deformation of it, which is negligibly different from any PPT adversary's view, does run in polynomial time.

2.3.7 One-Time Signature Schemes

So far we have presented different flavours of encryption schemes whose goals were to ensure the privacy of data. Digital signature schemes, on the other hand,

are concerned with the problem of *authenticity* of data, ensuring the receiver of the origin and integrity of data. A (digital) signature scheme is formally defined via three PPT algorithms are follows.

- $\mathbb{G}_{\mathtt{SIG}}(1^\kappa)$: This is the probabilistic key generation algorithm which, on input of the security parameter, outputs a verification/signing key pair $(\mathtt{vk}, \mathtt{sk})$. This algorithm might be split into a global parameter and a key generation algorithm as for PKE schemes.

- $\mathtt{Sig}(\mathtt{m}, \mathtt{sk})$: This is the probabilistic signature algorithm, which takes a message $\mathtt{m}$ and a secret key $\mathtt{sk}$, and returns a signature σ.

- $\mathtt{Ver}(\mathtt{m}, \sigma, \mathtt{vk})$: This is the deterministic verification algorithm which, given a message $\mathtt{m}$, a signature σ on the message and a verification key $\mathtt{vk}$, returns either 0 (for reject) or 1 (for accept).

In our constructions of workflow schemes in Chapter 6, we achieve chosen ciphertext security using an adaptation of the technique by Canetti et al. [29] which is based on a one-time signature (OTS) scheme. An OTS is a signature scheme which satisfies only a weak notion of security where the adversary gets to see only one signature. For a general signature scheme, however, the adversary has access to a signing oracle on messages of its choice, providing access to multiple signatures.

The *strong existential unforgeability* of an OTS scheme is defined through the following game in which any PPT adversary must have negligible advantage.

UF

1. $(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathbb{G}_{\mathtt{OTS}}(1^\kappa)$
2. $(\mathtt{m}, \mathrm{st}) \leftarrow A_1(\mathtt{vk})$
3. $\sigma \leftarrow \mathtt{Sig}(\mathtt{sk}, \mathtt{m})$
4. $(\mathtt{m}', \sigma') \leftarrow A_2(\sigma, \mathrm{st})$

The advantage is defined to be:

$$\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(A) := \Pr[(\mathtt{m}', \sigma') \neq (\mathtt{m}, \sigma) \wedge \mathtt{Ver}(\mathtt{m}', \sigma', \mathtt{vk}) = 1].$$

One-time signature schemes meeting this security definition can be constructed from any one-way function [61].

2.3.8 Secret Sharing Schemes

Informally, a secret sharing scheme allows a number of parties to share a secret between them in a way which allows the original secret to be reconstructed if, for example, a threshold number of parties provide their shares. In this thesis, we follow the approach in [19] for secret sharing over general access structures.

Definition 2.13. *A collection $\mathcal{P}$ of subsets of a set $P = \{X_1, \ldots, X_n\}$ is called a monotone access structure on P if:*

$$\forall A \in \mathcal{P} \text{ and } \forall B \subseteq P, A \subseteq B \Rightarrow B \in \mathcal{P}.$$

A set $Q \subseteq P$ is called a qualifying subset of P if $Q \in \mathcal{P}$.

The access structures considered in this work are all monotone and non-trivial ($\mathcal{P} \neq \emptyset$). Note that non-triviality implies $P \in \mathcal{P}$.

A secret sharing scheme is defined as a pair of algorithms as follows.

- $\mathbb{S}(1^\kappa, \mathtt{s}, \mathcal{P})$: This is the probabilistic secret sharing algorithm which on input of the security parameter 1^κ, a string $\mathtt{s}$ and a (monotone) access structure $\mathcal{P}$, outputs a list of shares $\mathbf{shr} = (\mathtt{shr}_1, \ldots, \mathtt{shr}_n)$ one for each element in $P = \{X_1, \ldots, X_n\}$ as well as some auxiliary information $\mathtt{aux}$.

- $\mathbb{S}^{-1}(\mathbf{shr}, \mathtt{aux})$: This is the deterministic secret reconstruction algorithm. On input a list of shares **shr** and some auxiliary information $\mathtt{aux}$, this algorithm

outputs a secret $\mathtt{s}$ or a failure symbol $\perp$.

A secret sharing scheme is *correct* if for all access structures $\mathcal{P}$ and strings $\mathtt{s} \in \{0,1\}^*$ of polynomial length in κ, the following experiment returns 1 with probability 1.

1. $(\mathbf{shr}, \mathtt{aux}) \leftarrow \mathbb{S}(1^\kappa, \mathtt{s}, \mathcal{P})$
2. $Q \leftarrow \mathcal{P}$
3. Parse $(X_{i_1}, \ldots, X_{i_k}) \leftarrow Q$
4. $[\mathbf{shr}']_j \leftarrow [\mathbf{shr}]_{i_j}, 1 \leq j \leq k$
5. $\mathtt{s}' = \mathbb{S}^{-1}(\mathbf{shr}', \mathtt{aux})$
6. If $\mathtt{s}' = \mathtt{s}$ return 1 else return 0

Our workflow schemes in Chapter 6 use secret sharing schemes as a building-block. The level of security provided by a secret sharing scheme will influence the overall security of our constructions. We consider both perfect and computational (non-perfect) secret sharing schemes [59].

For perfect secret sharing we will not require a game-based security definition. When necessary, we use an information-theoretical argument based on the following definition of security.

Definition 2.14 (Perfect Secret Sharing)**.** *A secret sharing scheme provides perfect secrecy if every non-qualifying subset of shares does not contain any information about the secret (in the information-theoretic sense). Formally, for any non-empty and non-qualifying set* $\{i_1, \ldots, i_n\}$ *of an access structure* $\mathcal{P}$*, and for every two secrets* sec_0 *and* sec_1*, let* $(\mathtt{aux}_b, \mathbf{shr}_b) \leftarrow \mathbb{S}(\mathrm{sec}_b, \mathcal{P})$*, for* $b \in \{0,1\}$*. Then, for every possible share value* $\mathtt{shr}_{i_j}$*,* $1 \leq j \leq n$ *and for every possible* $\mathtt{aux}$ *value*

$$\Pr[\mathtt{shr}_{i_j} = [\mathbf{shr}_0]_{i_j}] = \Pr[\mathtt{shr}_{i_j} = [\mathbf{shr}_1]_{i_j}] \text{ and } \Pr[\mathtt{aux} = \mathtt{aux}_0] = \Pr[\mathtt{aux} = \mathtt{aux}_1].$$

Note that, for perfect secret sharing schemes we do not have an asymptotic definition of security, and therefore we drop the security parameter in the primitive definition.

In perfect secret sharing schemes, the secret size constitutes a lower bound on the individual size of shares. To reduce this lower bound, one must relax the security definition and settle for polynomial-time indistinguishability. For computational secret sharing, we shall use the following definitions of security: secret indistinguishability against selective share attacks (IND-SSA), and against chosen share attacks (IND-CSA).

IND-SSA

1. $(\mathtt{s}_0, \mathtt{s}_1, \mathcal{P}^*, i_1, \ldots, i_k, \mathrm{st}) \leftarrow A_1(1^\kappa)$
2. $b \leftarrow \{0,1\}$
3. $(\mathbf{shr}^*, \mathtt{aux}^*) \leftarrow \mathbb{S}(1^\kappa, \mathtt{s}_b, \mathcal{P}^*)$
4. $b' \leftarrow A_2(\mathtt{aux}^*, ([\mathbf{shr}^*]_{i_j})_{j=1}^k, \mathrm{st})$

IND-CSA

1. $(\mathtt{s}_0, \mathtt{s}_1, \mathcal{P}^*, \mathrm{st}) \leftarrow A_1(1^\kappa)$
2. $b \leftarrow \{0,1\}$
3. $(\mathbf{shr}^*, \mathtt{aux}^*) \leftarrow \mathbb{S}(1^\kappa, \mathtt{s}_b, \mathcal{P}^*)$
4. $b' \leftarrow A_2^{\mathcal{O}}(\mathtt{aux}^*, \mathrm{st})$

In the SSA model, $k \leq n$, and $\{i_1, \ldots, i_k\}$ must not include a qualifying set of shares in $\mathcal{P}^*$. In the CSA model, $\mathcal{O}$ is a share extraction oracle subject to the condition that the adversary cannot extract a set of shares corresponding to a qualifying set in $\mathcal{P}^*$. The advantage is:

$$\mathrm{Adv}_{\mathtt{SS}}^{\mathtt{IND-atk}}(A) := |2\Pr[b' = b] - 1|,$$

where $\mathtt{atk} \in \{\mathtt{SSA}, \mathtt{CSA}\}$.

2.4 Cryptographic Constructions

For completeness, in this section we describe the details of some well-known cryptographic constructions which are referred to frequently in this work.

2.4.1 The KEM-DEM Construction

The formal details for the KEM-DEM construction, as discussed in Section 1.3, are shown below.

$\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK})$

- $(\mathtt{k}, \mathtt{c}_1) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$
- $\mathtt{c}_2 \leftarrow \mathbb{E}_{\mathtt{DEM}}(\mathtt{m}, \mathtt{k})$
- $\mathtt{c} \leftarrow (\mathtt{c}_1, \mathtt{c}_2)$
- Return $\mathtt{c}$

$\mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$

- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathtt{c}$
- $\mathtt{k} \leftarrow \mathbb{D}_{\mathtt{KEM}}(\mathtt{c}_2, \mathtt{SK})$
- If $\mathtt{k} = \perp$ return $\perp$
- $\mathtt{m} \leftarrow \mathbb{D}_{\mathtt{DEM}}(\mathtt{c}_2, \mathtt{k})$
- Return $\mathtt{m}$

The proof of the following theorem can be found in [37].

Theorem 2.1 (Cramer and Shoup)**.** *The above construction results in an IND-CCA secure public-key encryption scheme if the underlying KEM and DEM are IND-CCA and FG-CCA secure respectively. More precisely, for any adversary A there exist adversaries B_1 and B_2 such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}}(A) \leq 2\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{KEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2).$$

2.4.2 The Fujisaki–Okamoto Transformation

Fujisaki and Okamoto [45] gave a general transformation which converts any (reasonable) OW-CPA secure public-key encryption scheme to one which achieves IND-CCA security in the random oracle model. This transformation was later extended to incorporate identity-based and certificateless encryption schemes [89, 63].

The key generation algorithm of the transformed scheme is the same as that in the underlying PKE scheme. Encryption and decryption are as follows.

$\mathbb{E}_{\mathtt{PKE}'}(\mathtt{m}, \mathtt{PK})$

- $\mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$
- $r \leftarrow H(\mathtt{m}', \mathtt{m})$
- $\mathtt{c}_1 \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}', \mathtt{PK}; r)$
- $\mathtt{c}_2 \leftarrow \mathbb{E}_{\mathtt{SKE}}(\mathtt{m}, \mathrm{KDF}(\mathtt{m}'))$
- $\mathtt{c} \leftarrow (\mathtt{c}_1, \mathtt{c}_2)$
- Return $\mathtt{c}$

$\mathbb{D}_{\mathtt{PKE}'}(\mathtt{c}, \mathtt{SK})$

- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathtt{c}$
- $\mathtt{m}' \leftarrow \mathbb{D}_{\mathtt{PKE}}(\mathtt{c}_1, \mathtt{SK})$
- If $\mathtt{m}' = \perp$ return $\perp$
- $\mathtt{m} \leftarrow \mathbb{D}_{\mathtt{SKE}}(\mathtt{c}_2, \mathrm{KDF}(\mathtt{m}'))$
- If $\mathtt{m} = \perp$ return $\perp$
- $r \leftarrow H(\mathtt{m}', \mathtt{m})$
- If $\mathtt{c}_1 \neq \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}', \mathtt{PK}; r)$ return $\perp$
- Return $\mathtt{m}$

Here the key derivation function $\mathrm{KDF} : \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK}) \rightarrow \mathcal{K}_{\mathtt{SKE}}(\kappa)$ is a cryptographic hash function. It is proved in [45] that:

Theorem 2.2 (Fujisaki and Okamoto)**.** *In the random oracle model, the above transformation results in an IND-CCA public-key encryption scheme if the underlying PKE scheme is OW-CPA secure and γ-uniform and the SKE is FG-CCA secure. More precisely, for any adversary A there exist adversaries B_1 and B_2 such that* $\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}'}(A)$ *is at most*

$$(2(q_H + q_K)\varepsilon_1 + \varepsilon_2 + 1)\left(1 - 2\varepsilon_1 - 2\varepsilon_2 - \frac{\gamma(\mathtt{PK})}{|\mathcal{R}_{\mathtt{PKE}}(\mathtt{PK})|} - \frac{1}{|\mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})|}\right)^{-q_D} - 1,$$

where $\varepsilon_1 = \mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{PKE}}(B_1)$, $\varepsilon_2 = \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{SKE}}(B_2)$, q_H, q_K *and* q_D *denote the number of queries to H,* KDF *and decryption queries respectively and* $\gamma(\mathtt{PK})$ *is as defined in Section 2.3.2.*

A refined security analysis of this transformation is presented in [47].

2.4.3 A Generic Construction of KEMs

Dent [38] describes a method akin to that introduced by Fujisaki and Okamoto [45] which builds an IND-CCA secure KEM using an OW-CPA secure PKE scheme. Combined with a secure DEM, we obtain an alternative way to build a secure PKE scheme.

$\mathbb{E}_{\mathtt{KEM}}(\mathtt{PK})$

- $\mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})$
- $r \leftarrow H(\mathtt{m}')$
- $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}', \mathtt{PK}; r)$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\mathtt{m}')$
- Return $(\mathtt{k}, \mathtt{c})$

$\mathbb{D}_{\mathtt{KEM}}(\mathtt{c}, \mathtt{SK})$

- $\mathtt{m}' \leftarrow \mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$
- If $\mathtt{m}' = \perp$ return $\perp$
- $r \leftarrow H(\mathtt{m}')$
- If $\mathtt{c} \neq \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}', \mathtt{PK}; r)$ return $\perp$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\mathtt{m}')$
- Return $\mathtt{k}$

Theorem 2.3 (Dent)**.** *In the random oracle model, the above transformation results in an IND-CCA secure key encapsulation mechanism if the underlying PKE scheme is OW-CPA secure and γ-uniform. More precisely, for any adversary A there exists an adversary B such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}'}(A) \leq 2(q_D + q_H + q_K)\mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{PKE}}(B) + \frac{2q_D}{|\mathcal{M}_{\mathtt{PKE}}(\mathtt{PK})|} + \frac{2\gamma(\mathtt{PK})q_D}{|\mathcal{R}_{\mathtt{PKE}}(\mathtt{PK})|},$$

where q_H, q_K and q_D are the number of queries to H, KDF *and the decapsulation oracles respectively and $\gamma(\mathtt{PK})$ is as defined in Section 2.3.2.*

The transparency of security proofs is often cited as a problem in the cryptographic community. The proofs of Theorems 2.1 and 2.3 are considerably simpler than that of Theorem 2.2 given in [45]. This demonstrates one of the benefits of a modular approach to design of cryptosystems.

2.4.4 The ElGamal Encryption Scheme

Given a computational group scheme Γ with a generator P of order p, the ElGamal public-key encryption scheme [44] operates as follows. The global parameter is $I = \Gamma$ which includes the security parameter and the group definition. The key generation algorithm $\mathbb{G}_{\mathtt{PKE}}(I)$ takes the global parameters and returns a key pair $(\mathtt{SK}, \mathtt{PK}) = (x, xP)$ for $x \leftarrow \mathbb{Z}_p^*$. The encryption algorithm $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r)$ takes the public key and a message in the message space $\mathcal{M}_{\mathtt{PKE}}(I) = G$, and returns a ciphertext $(\mathtt{c}_1, \mathtt{c}_2) := (rP, \mathtt{m} + r \cdot \mathtt{PK})$ for $r \leftarrow \mathbb{Z}_p^*$. The decryption algorithm $\mathbb{D}_{\mathtt{PKE}}((\mathtt{c}_1, \mathtt{c}_2), \mathtt{SK})$ takes a ciphertext in $G \times G$ and a secret key and returns the message $\mathtt{m} = \mathtt{c}_2 - \mathtt{SK} \cdot \mathtt{c}_1$. This scheme could be easily shown to be IND-CPA secure under the DDH assumption [87]:

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(A) \leq 2\mathrm{Adv}^{\mathtt{DDH}}_{\Gamma}(B).$$

The "hashed" version of this scheme, where $\mathtt{c}_2 = \mathtt{m} \oplus H(r\mathtt{PK})$, can be shown to be IND-CPA secure under the weaker CDH assumption in the random oracle model. Hashed ElGamal KEM [37] is shown to be IND-CCA secure under the gap Diffie–Hellman assumption in the ROM.

2.4.5 The IBE Scheme of Boneh and Franklin

Boneh and Franklin [23] gave two identity-based encryption schemes, one called BasicIdent satisfying the OW-CPA security definition and the other called FullIdent satisfying the IND-CCA security definition of an IBE scheme. In one of our constructions in Chapter 3 we will use the BasicIdent system, but will then compare the construction with the FullIdent system.

For a computational bilinear group scheme Γ, we will need to define crypto-

graphic hash functions:

$$\begin{aligned} H_1 &: \{0,1\}^* \longrightarrow G_2, \\ H_2 &: G_T \longrightarrow \{0,1\}^n, \\ H_3 &: \{0,1\}^* \longrightarrow \mathbb{Z}_p^*, \end{aligned}$$

for an integer n corresponding to the length of messages to be encrypted.

In both schemes the trusted authority's keys are given by $\mathtt{Mpk} = \mathtt{Msk} \cdot P_1$ for $\mathtt{Msk} \in \mathbb{Z}_p^*$ chosen uniformly at random, and the user private key D for identity $\mathtt{ID}$ is $D = \mathtt{Msk} \cdot Q_{\mathtt{ID}}$ where $Q_{\mathtt{ID}} := H_1(\mathtt{ID})$.

BasicIdent:

We define $\mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk}) = \{0,1\}^n$, $\mathcal{C}_{\mathtt{IBE}}(\mathtt{Mpk}) = G_1 \times \{0,1\}^n$ and $\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk}) = \mathbb{Z}_p^*$. We then define

$\mathbb{E}_{\mathtt{IBE}}^{\mathtt{BasicIdent}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$

- $\mathtt{c}_1 \leftarrow rP_1$
- $Q_{\mathtt{ID}} \leftarrow H_1(\mathtt{ID})$
- $t \leftarrow e(\mathtt{Mpk}, Q_{\mathtt{ID}})^r$
- $\mathtt{c}_2 \leftarrow \mathtt{m} \oplus H_2(t)$
- Return $(\mathtt{c}_1, \mathtt{c}_2)$

$\mathbb{D}_{\mathtt{IBE}}^{\mathtt{BasicIdent}}(\mathtt{c}, D)$

- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathtt{c}$
- $t \leftarrow e(U, D)$
- $\mathtt{m} \leftarrow \mathtt{c}_2 \oplus H_2(t)$
- Return $\mathtt{m}$

Note that $\gamma(\mathtt{Mpk})/|\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})|$ for BasicIdent is equal to $1/p \approx 2^{-\kappa}$.

Boneh and Franklin prove the following security result about BasicIdent.

Theorem 2.4. *Let H_1 and H_2 be modelled as random oracles and suppose A is an adversary against BasicIdent making at most q_X private key extraction queries and q_2 queries of H_2. Then there is an algorithm B with essentially the same running*

time of A such that

$$\mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{IBE}}(A) \leq e \cdot (1+q_X) \cdot q_2 \cdot \left(\mathrm{Adv}^{\mathtt{CBDH}}_{\Gamma}(B) + \frac{1}{q_2 \cdot 2^n} \right).$$

Here, and in Theorems 2.5 and 2.6 below, e is the base of the natural logarithm.

FullIdent:

Although not explicitly defined in [23], there are in fact two variants of FullIdent mentioned in [23]. We shall present both here.

FullIdent-1:

We define $\mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk}) = \{0,1\}^n$, $\mathcal{C}_{\mathtt{IBE}}(\mathtt{Mpk}) = G_1 \times \{0,1\}^n \times \{0,1\}^{|\mathtt{m}|}$, $\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk}) = \{0,1\}^n$. We require all the hash functions used by BasicIdent and in addition we require a cryptographic hash function $H_4 : \{0,1\}^n \longrightarrow \{0,1\}^{|\mathtt{m}|}$, to encrypt messages of length $|\mathtt{m}|$. We define the scheme as follow.

$\mathbb{E}^{\mathtt{FullIdent-1}}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; \sigma)$

- $r \leftarrow H_3(\sigma, \mathtt{m})$
- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathbb{E}^{\mathtt{BasicIdent}}_{\mathtt{IBE}}(\sigma, \mathtt{ID}, \mathtt{Mpk}; r)$
- $\mathtt{c}_3 \leftarrow \mathtt{m} \oplus H_4(\sigma)$
- Return $(\mathtt{c}_1, \mathtt{c}_2, \mathtt{c}_3)$

$\mathbb{D}^{\mathtt{FullIdent-1}}_{\mathtt{IBE}}(\mathtt{c}, D)$

- $(\mathtt{c}_1, \mathtt{c}_2, \mathtt{c}_3) \leftarrow \mathtt{c}$
- $\sigma \leftarrow \mathbb{D}^{\mathtt{BasicIdent}}_{\mathtt{IBE}}((\mathtt{c}_1, \mathtt{c}_2), D)$
- $\mathtt{m} \leftarrow \mathtt{c}_3 \oplus H_4(\sigma)$
- $r \leftarrow H_3(\sigma, \mathtt{m})$
- If $rP_1 \neq \mathtt{c}_1$ return $\perp$
- Return $\mathtt{m}$

The authors in [23] handle the security of the FullIdent-1 scheme above by first reducing it to the security of a standard public-key encryption and then establishing the security of this latter scheme using the Fujisaki–Okamoto transformation as described in Section 2.4.2.

Theorem 2.5. *Let H_1, H_2, H_3 and H_4 be modelled as random oracles and suppose A is an adversary against FullIdent-1 making at most q_X private key extraction queries, q_D decryption queries and q_2, q_3 and q_4 queries of H_2, H_3 and H_4 respectively. Then there is an algorithm B, running in time essentially that of A, such that*

$$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-CCA}}(A) \leq c_1 \cdot \left(\frac{1}{c_2} \left(c_3 \cdot \left(q_2 \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(B) + \frac{1}{2^n} \right) + 1 \right) - 1 \right),$$

where $c_1 = e \cdot (1 + q_X + q_D)$, $c_2 = (1 - 2/p)^{q_D}$ and $c_3 = 2 \cdot (q_3 + q_4)$.

FullIdent-2:

Let $(\mathbb{G}_{\mathtt{SKE}}, \mathbb{E}_{\mathtt{SKE}}, \mathbb{D}_{\mathtt{SKE}})$ denote a symmetric encryption scheme, with message space $\mathcal{M}_{\mathtt{SKE}}(\kappa)$, ciphertext space $\mathcal{C}_{\mathtt{SKE}}(\kappa)$ and key space $\mathcal{K}_{\mathtt{SKE}}(\kappa)$ for security parameter κ. We assume that this symmetric algorithm is secure in the sense of FG-CPA. The following scheme is mentioned but not analysed in [23]. We define

$$\begin{aligned} \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk}) &= \mathcal{M}_{\mathtt{SKE}}(\kappa) \\ \mathcal{C}_{\mathtt{IBE}}(\mathtt{Mpk}) &= G_1 \times \{0,1\}^n \times \mathcal{C}_{\mathtt{SKE}}(\kappa) \\ \mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk}) &= \{0,1\}^n. \end{aligned}$$

We also let

$$H_5 : \{0,1\}^* \longrightarrow \mathcal{K}_{\mathtt{SKE}}(\kappa)$$

denote a cryptographic hash function. We then define encryption and decryption algorithms as follows.

$\mathbb{E}^{\texttt{FullIdent}-2}_{\texttt{IBE}}(\texttt{m}, \texttt{ID}, \texttt{Mpk}; \sigma)$

- $r \leftarrow H_3(\sigma, \texttt{m})$
- $(\texttt{c}_1, \texttt{c}_2) \leftarrow \mathbb{E}^{\texttt{BasicIdent}}_{\texttt{IBE}}(\sigma, \texttt{ID}, \texttt{Mpk}; r)$
- $\texttt{c}_3 \leftarrow \mathbb{E}_{\texttt{SKE}}(\texttt{m}, H_5(\sigma))$
- Return $(\texttt{c}_1, \texttt{c}_2, \texttt{c}_3)$

$\mathbb{D}^{\texttt{FullIdent}-2}_{\texttt{IBE}}(\texttt{c}, D)$

- $(\texttt{c}_1, \texttt{c}_2, \texttt{c}_3) \leftarrow \texttt{c}$
- $\sigma \leftarrow \mathbb{D}^{\texttt{BasicIdent}}_{\texttt{IBE}}((\texttt{c}_1, \texttt{c}_2), D)$
- $\texttt{m} \leftarrow \mathbb{D}_{\texttt{SKE}}(\texttt{c}_3, H_5(\sigma))$
- $r \leftarrow H_3(\sigma, \texttt{m})$
- If $rP_1 \neq \texttt{c}_1$ return $\perp$
- Return $\texttt{m}$

The following result can be proved for the scheme FullIdent-2 using the same argument as is used in [23] for FullIdent-1. Alternatively one could use the result in [89] to derive this result.

Theorem 2.6. *Suppose H_1, H_2, H_3 and H_5 are modelled as random oracles and A is an adversary against FullIdent-2 making at most q_X private key extraction queries, q_D decryption queries and q_2, q_3 and q_5 queries of H_2, H_3 and H_5 respectively. Then there are algorithms B_1 and B_2, running in time essentially that of A, such that*

$$\mathrm{Adv}^{\texttt{IND}-\texttt{CCA}}_{\texttt{IBE}}(A) \leq c_1 \cdot \left(\frac{1}{c_2} \left(\begin{array}{r} c_3 \cdot \left(q_2 \cdot \mathrm{Adv}^{\texttt{CBDH}}_{\Gamma}(B_1) + \frac{1}{2^n} \right) \\ + \mathrm{Adv}^{\texttt{FG}-\texttt{CPA}}_{\texttt{SKE}}(B_2) + 1 \end{array} \right) - 1 \right),$$

where $c_1 = e \cdot (1 + q_X + q_D)$, $c_3 = 2 \cdot (q_3 + q_5)$ and

$$c_2 = \left(1 - 2 \left(q_2 \cdot \mathrm{Adv}^{\texttt{CBDH}}_{\Gamma}(B_1) + \frac{1}{2^n} \right) - 2\mathrm{Adv}^{\texttt{FG}-\texttt{CPA}}_{\texttt{SKE}}(B_2) - \frac{1}{p} - \frac{1}{|\mathcal{M}_{\texttt{SKE}}(\texttt{Mpk})|} \right)^{q_D}.$$

2.4.6 The IBE Scheme of Sakai and Kasahara

Sakai and Kasahara [77] gave another identity-based encryption scheme based on bilinear maps. The key generation algorithm of this scheme is identical to that in

the IBE scheme of Boneh and Franklin: a random element of $\mathbb{Z}_p^*$ is the master secret key and the master public key is set to be $\mathtt{Msk} \cdot P_1$. The user private keys are derived using the following alternative method:

$$\mathbb{X}_{\mathtt{IBE}}^{\mathtt{BasicSK}}(\mathtt{ID}, \mathtt{Msk}) := \frac{1}{\mathtt{Msk} + H_1(\mathtt{ID})} P_2,$$

where $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$ is a cryptographic hash function. The encryption and decryption algorithms are:

$\mathbb{E}_{\mathtt{IBE}}^{\mathtt{BasicSK}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$

- $r \leftarrow \mathbb{Z}_p^*$
- $\mathtt{c}_1 \leftarrow r(\mathtt{Mpk} + H_1(\mathtt{ID}) \cdot P_1)$
- $\mathtt{c}_2 \leftarrow \mathtt{m} \oplus H_2(e(P_1, P_2)^r)$
- $\mathtt{c} \leftarrow (\mathtt{c}_1, \mathtt{c}_2)$
- Return $\mathtt{c}$

$\mathbb{D}_{\mathtt{IBE}}^{\mathtt{BasicSK}}(\mathtt{c}, D)$

- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathtt{c}$
- $\mathtt{m} \leftarrow \mathtt{c}_2 \oplus H_2(e(\mathtt{c}_1, D))$
- Return $\mathtt{m}$

Note that if $e(P_1, P_2)$ is pre-computed, no pairing computations will be needed when encrypting data. Note also that no hashing into group G_2 is needed.

Chen et al. in [30] established the IND-CPA security of this construction under the q-BDHI assumption (see Section 2.2.2). The authors then use the Fujisaki–Okamoto transformation a la Boneh and Franklin to build a fully secure IND-CCA scheme in the random oracle model.

Chapter 3

Identity-Based Key Encapsulation Mechanisms

In this chapter we extend the concept of key encapsulation mechanisms to the identity-based setting (an IBKEM). After defining appropriate security models for this primitive, we show that the natural combination of secure IBKEMs and data encapsulation mechanisms results in secure identity-based encryption schemes. Next, we give a generic construction of IBKEMs, and compare a specific instantiation of it with the FullIdent-2 scheme (Section 2.4.5) and a construction due to Lynn [66], whose security is proved in Section 3.2.1. Parts of this chapter, first appeared in [20] and represent joint work with Kamel Bentahar, John Malone-Lee and Nigel Smart.

3.1 The IBKEM-DEM Paradigm

3.1.1 The IBKEM Primitive

An IBKEM is specified by four PPT algorithms as follows.

- $\mathbb{G}_{\mathtt{IBKEM}}(1^\kappa)$. A probabilistic algorithm which takes as input 1^κ and returns

the master public key $\mathtt{Mpk}$ and the master secret key $\mathtt{Msk}$. This algorithm also outputs a description of the key space. The remark regarding the key generation algorithm of a PKE made in Section 2.3.2 also applies here.

- $\mathbb{X}_{\mathtt{IBKEM}}(\mathtt{ID},\mathtt{Msk})$. A (possibly) probabilistic algorithm which takes as input $\mathtt{Msk}$ and an identifier string $\mathtt{ID} \in \{0,1\}^*$ for a user and returns the associated private key D.

- $\mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID},\mathtt{Mpk})$. This is the probabilistic encapsulation algorithm. On input of $\mathtt{ID}$ and $\mathtt{Mpk}$ this outputs a pair $(\mathtt{k},\mathtt{c})$ where $\mathtt{k} \in \mathcal{K}_{\mathtt{IBKEM}}(\mathtt{Mpk})$ is a key and $\mathtt{c} \in \mathcal{C}_{\mathtt{IBKEM}}(\mathtt{Mpk})$ is an encapsulation of that key.

- $\mathbb{D}_{\mathtt{IBKEM}}(\mathtt{c},D)$. This is the deterministic decapsulation algorithm. On input an encapsulation $\mathtt{c}$ and a user secret key D this algorithm outputs either a key $\mathtt{k}$ or a failure symbol $\perp$.

The security of an IBKEM is defined through the following two-stage games between an adversary A and a challenger.

OW-atk

1. $(\mathtt{Msk},\mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBKEM}}(1^\kappa)$
2. $(\mathtt{ID}^*,\mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk})$
3. $(\mathtt{k},\mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID}^*,\mathtt{Mpk})$
4. $\mathtt{k}' \leftarrow A_2^{O_2}(\mathtt{c}^*,\mathrm{st})$

$\mathrm{Adv}^{\mathtt{OW-atk}}_{\mathtt{IBKEM}}(A) = \Pr[\mathtt{k}' = \mathtt{k}]$.

IND-atk

1. $(\mathtt{Msk},\mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBKEM}}(1^\kappa)$
2. $(\mathtt{ID}^*,\mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk})$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{IBKEM}}(\mathtt{Mpk})$
4. $(\mathtt{k}_1,\mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID}^*,\mathtt{Mpk})$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b,\mathtt{c}^*,\mathrm{st})$

$\mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{IBKEM}}(A) = |2\Pr[b' = b] - 1|$.

In the above O_i denotes oracles to which the adversary has access. There are two possibilities for these oracles depending on the attack model for our game:

- In the CPA model (atk=CPA) the adversary only has access to a private key extraction oracle which, on input of $\mathtt{ID} \neq \mathtt{ID}^*$, will output the corresponding value of D.
- In the CCA model (atk=CCA) the adversary has access not only to a private key extraction oracle, but also to a decapsulation oracle with respect to any identity $\mathtt{ID}$ of the adversary's choice. The adversary has access to this decapsulation oracle, subject to the restriction that in the second phase it is not allowed to call it with the pair $(\mathtt{c}^*, \mathtt{ID}^*)$.

An IBKEM is considered to be secure, in the sense of a given goal and attack model (IND-CCA for example) if for all PPT adversaries A, the advantage in the relevant game above is a negligible function of the security parameter κ. In this thesis, we shall be concerned with IND-CCA secure IBKEMs and have presented the one-way game for completeness only.

3.1.2 Combining IBKEMs with DEMs

In order to apply our IBKEM constructions we will need to combine them with data encapsulation mechanisms. In addition, when we compare our IBKEM-DEM construction with that of the original Boneh–Franklin identity-based encryption scheme, we will need to know the exact security guarantees.

We assume that the key space output by the IBKEM corresponds to the key space required by the DEM. Our construction follows that in [37, Section 7.3] and consists of the natural concatenation of the key encapsulation followed by the data encapsulation of the message under the key encapsulated by the first component. We denote such a ciphertext $\mathtt{c} = (\mathtt{c}_1, \mathtt{c}_2)$ henceforth, where $\mathtt{c}_1$ encapsulates the key and $\mathtt{c}_2$ encapsulates the data, and we refer to such a construction as *hybrid*.

The following theorem is a natural generalisation of Theorem 2.1. Note that, unlike the equivalent result in [37], we are implicitly assuming that for all keys,

all encapsulations decapsulate properly. As noted in Section 2.3.2, it would be straightforward to generalise the result; however, the correctness condition that we are assuming applies to all the primitives that we consider in this thesis.

Theorem 3.1. *Let A be a PPT adversary against the hybrid identity-based encryption scheme in the IND-CCA sense. Then there exists PPT adversaries B_1 and B_2, whose running times are essentially that of A, such that*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(A) \leq 2\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2).$$

We defer the proof of this theorem until Section 4.2 where we shall prove a more general result.

3.2 IBKEM Constructions

In this section, we present three constructions of IBKEMs. The first two are based on specific computational problems, namely variants of the bilinear Diffie–Hellman problem on elliptic curves as described in Section 2.2.2, while the third construction is generic.

The first construction is due to Lynn [66]. We note that Lynn only mentions this IBKEM construction in passing and does not give a security definition or proof. In Section 3.2.1 we present a security proof for this construction under the GBDH problem (see Section 2.2.2). The second construction is a variant of Lynn's construction and is based on Dent's fifth generic construction described in Section 2.4.3. It is secure under the standard CBDH problem. Our third construction takes any probabilistic OW-CPA secure identity-based encryption scheme and produces an IND-CCA secure IBKEM.

The second construction can be considered as either a strengthening of Lynn's construction or as an optimisation of the third generic construction, when in the

third construction we use the BasicIdent algorithm from Section 2.4.5. The second construction is less efficient than Lynn's method, but its security rests on the CBDH problem rather than the stronger GBDH problem. Since the CBDH problem is a more natural and well studied problem than the GBDH problem, we see this extra confidence in the security as more than offsetting the slight performance reduction compared to Lynn's construction.

Our first two constructions follow the standard setup for identity-based systems built on pairings, as first explained in [23]. For completeness, we recall the setup here. We let H_1, H_2, H_3, H_4 and H_5 denote cryptographic hash functions as described in Section 2.4.5. However, now we let $(\mathbb{G}_{\mathtt{DEM}}, \mathbb{E}_{\mathtt{DEM}}, \mathbb{D}_{\mathtt{DEM}})$ denote a DEM, and so the co-domain of H_5 is the key space of the DEM.

For our first two constructions we adopt the initial setup as in the Boneh–Franklin IBE scheme by defining the trusted authority's keys as $\mathtt{Mpk} = \mathtt{Msk} \cdot P_1$ for $\mathtt{Msk} \in \mathbb{Z}_p^*$ chosen uniformly at random. The user private key D corresponding to identity $\mathtt{ID}$ is $D = \mathtt{Msk} \cdot H_1(\mathtt{ID})$. We sometimes write $Q_{\mathtt{ID}}$ for the point (on an elliptic curve) that we obtain by computing $H_1(\mathtt{ID})$.

3.2.1 Lynn's IBKEM

The scheme proposed by Lynn [66] is given below.

$\mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID}, \mathtt{Mpk})$

- $r \leftarrow \mathbb{Z}_p^*$
- $\mathtt{c} \leftarrow rP_1$
- $T \leftarrow e(\mathtt{Mpk}, H_1(\mathtt{ID}))^r$
- $\mathtt{k} \leftarrow H_5(T)$
- Return $(\mathtt{k}, \mathtt{c})$

$\mathbb{D}_{\mathtt{IBKEM}}(\mathtt{c}, D)$

- $T \leftarrow e(\mathtt{c}, D)$
- $\mathtt{k} \leftarrow H_5(T)$
- Return $\mathtt{k}$

We now prove the security of this construction.

Theorem 3.2. *If the GBDH problem is hard and H_5 and H_1 are modelled as random oracles then the above construction is secure against adaptive chosen ciphertext attacks. Specifically, if A is a PPT algorithm that breaks the above IBKEM in the IND-CCA sense, then there exists a PPT algorithm B such that*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(A) \leq 2q_T \cdot \mathrm{Adv}^{\mathtt{GBDH}}_{\Gamma}(B, q_D^2 + 2q_5q_D + q_5) + \frac{2q_5}{p},$$

where $q_T = q_1 + q_X + q_D + 1$, and q_1, q_5, q_D, and q_X are the maximum number of queries that can be made by A to H_1, H_5, the decryption and the private key extraction oracles respectively.

Proof. Let S be the event that A correctly determines the bit chosen by the challenge encryption oracle during its attack. Let $\mathtt{ID}^*$ be the identity chosen by A during its attack and let T^* be the output of the bilinear map generated by A's challenge encapsulation oracle at the third stage of the encapsulation process. We define the following events.

- $\mathtt{AskK}$: The event that A makes the query T^* to H_5 during its attack.
- $\mathtt{AskH}$: The event that A makes the query $\mathtt{ID}^*$ to H_1 during its attack or that it makes a decryption query involving $\mathtt{ID}^*$.

We have

$$\begin{aligned} \Pr[S] &= \Pr[S \wedge \mathtt{AskK}] + \Pr[S \wedge \neg \mathtt{AskK}] \\ &= \Pr[S \wedge \mathtt{AskK}] + \frac{1}{2}. \end{aligned} \tag{3.1}$$

This follows from the fact that if A does not query T^* to H_5 then it can have no

advantage. We can then express

$$\begin{aligned}\Pr[S \wedge \mathtt{AskK}] &= \Pr[S \wedge \mathtt{AskK} \wedge \mathtt{AskH}] + \Pr[S \wedge \mathtt{AskK} \wedge \neg\mathtt{AskH}] \\ &\leq \Pr[S \wedge \mathtt{AskK} \wedge \mathtt{AskH}] + \frac{q_5}{p}. \end{aligned} \tag{3.2}$$

Here the inequality comes from the fact that, if A does not make the query $\mathtt{ID}^*$ to H_1, then $Q_{\mathtt{ID}^*}$ is completely unknown to A and hence, from A's view, there are p equally likely possibilities for T^*.

Using the definition of A's advantage with (3.1) and (3.2) we have

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(A) \leq 2\Pr[S \wedge \mathtt{AskK} \wedge \mathtt{AskH}] + \frac{2q_5}{p}. \tag{3.3}$$

To complete the proof we argue that, in the event $S \wedge \mathtt{AskK} \wedge \mathtt{AskH}$, there is an adversary B that solves the GBDH problem with probability at least $1/q_T$, where q_T is as in the statement of the theorem, while making at most $q_D^2 + 2q_5q_D + q_5$ queries to its DBDH oracle.

Suppose that we have a CBDH problem instance (aP_2, bP_2, cP_2) that we wish to solve. We construct an algorithm B to do this by using A. We set the value of $\mathtt{Mpk}$ for algorithm A to be $cP_1 = \rho(cP_2)$, where ρ is the isomorphism described in Section 2.2.2.

We maintain three lists $L_1 \subset \{0,1\}^* \times G_2 \times \mathbb{Z}_p$, $L_5 \subset G_T \times \mathcal{K}_{\mathtt{DEM}}(\kappa)$ and $L_D \subset G_1 \times G_2 \times G_1 \times \mathcal{K}_{\mathtt{DEM}}(\kappa)$ that allow us to provide consistent answers to A's oracle calls. We simulate these oracles as follows.

- H_1 **Queries**: We choose an index i from $\{1, \ldots, q_T\}$. We respond to queries to H_1 as follows (assuming that the query is not already present in L_1).
  - If we are responding to the i-th query, which we denote $\mathtt{ID}'$, we respond with bP_2 and add $(\mathtt{ID}', bP_2, \perp)$ to L_1.

  - If we are responding to any other query $\mathtt{ID}$ we choose $x \leftarrow \{1,\ldots,p\}$ and respond with xP_2. We also add $(\mathtt{ID}, xP_2, x)$ to L_1.

- H_5 **Queries**: We respond to a (non-repeat) query T to H_5 as follows. We first search L_D for an entry $(\mathtt{c}, bP_2, cP_1, \mathtt{k})$ such that the DBDH oracle returns 1 when queried with $(\mathtt{c}, bP_2, cP_1, T)$. If such an entry is found we respond with $\mathtt{k}$. Otherwise, we call the DBDH oracle with (aP_1, bP_2, cP_1, T). If the oracle returns 1 we output T and terminate the simulation – the solution to the CBDH problem has been found. If the oracle returns 0 we simply choose a random response $\mathtt{k}_T$ to respond with and update L_5 by adding $(T, \mathtt{k}_T)$.

- $\mathbb{X}_{\mathtt{IBKEM}}$ **Queries**: We respond to an extraction oracle query $\mathtt{ID}$ made by A as follows. We first call the H_1 oracle with $\mathtt{ID}$. We then search the list L_1 for the entry corresponding to $\mathtt{ID}$. If this entry has third component equal to $\perp$ then we abort. Otherwise we obtain the triple $(\mathtt{ID}, xP_2, x)$. We then respond with $D \leftarrow x(cP_2)$.

- $\mathbb{D}_{\mathtt{IBKEM}}$ **Queries**: We respond to a decapsulation oracle query $(\mathtt{c}, \mathtt{ID})$ as follows. We first call the oracle H_1. We then look for the entry corresponding to $\mathtt{ID}$ in L_1. There are two possibilities.

  - If $\mathtt{ID} = \mathtt{ID}'$, so the response we get from H_1 is bP_2. We then search L_5 for an element $(T, \mathtt{k}_T)$ such that the DBDH oracle returns 1 when queried with $(\mathtt{c}, bP_2, cP_1, T)$. If such an element is found we respond with $\mathtt{k}_T$. If no such element is found we choose a response $\mathtt{k}$ at random and add $(\mathtt{c}, bP_2, cP_1, \mathtt{k})$ to L_D.
  - If $\mathtt{ID} \neq \mathtt{ID}'$, we obtain the entry corresponding entry $(\mathtt{ID}, xP_2, x)$ from L_1 and we compute $D \leftarrow x(cP_2)$. We then compute $T \leftarrow e(\mathtt{c}, D)$ and call H_5 with T. We relay the response to A.

- **Challenge Query**: When A makes its challenge encryption oracle query we choose $\mathtt{k}'$ at random and we set $\mathtt{c}^* \leftarrow aP_1$. We respond with $(\mathtt{k}', \mathtt{c}^*)$.

In the above simulation what we want is that $H_1(\mathtt{ID}^*) = bP_1$. The oracle H_1 is called at most q_T times in the simulation above; therefore, we achieve this with probability at least $1/q_T$ in the event $\mathtt{AskH}$. Since by construction we also have $\mathtt{c}^* = aP_1$ and $\mathtt{Mpk} = cP_1$ we conclude that, in the event $\mathtt{AskH}$, with probability at least $1/q_T$, the value of T^* implicit in the challenge ciphertext is $e(P_1, P_2)^{abc}$; this is the value which we wish to compute. We conclude that

$$\Pr[S \wedge \mathtt{AskK} \wedge \mathtt{AskH}] \leq q_T \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B, q_D^2 + 2q_5 q_D + q_5)). \tag{3.4}$$

The result follows from (3.3) and (3.4). □

3.2.2 An IBKEM Based on BasicIdent

In Section 2.4.5 we presented the details of the BasicIdent encryption scheme. We now present an IBKEM based on this construction.

$\mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID}, \mathtt{Mpk})$

- $\mathtt{m} \leftarrow \{0,1\}^n$
- $r \leftarrow H_3(\mathtt{m})$
- $\mathtt{c}_1 \leftarrow rP_1$
- $\mathtt{c}_2 \leftarrow \mathtt{m} \oplus H_2(e(\mathtt{Mpk}, H_1(\mathtt{ID}))^r)$
- $\mathtt{c} \leftarrow (\mathtt{c}_1, \mathtt{c}_2)$
- $\mathtt{k} \leftarrow H_5(\mathtt{m})$
- Return $(\mathtt{k}, \mathtt{c})$

$\mathbb{D}_{\mathtt{IBKEM}}(\mathtt{c}, D)$

- $(\mathtt{c}_1, \mathtt{c}_2) \leftarrow \mathtt{c}$
- $\mathtt{m} \leftarrow V \oplus H_2(e(\mathtt{c}_1, \mathtt{c}_2))$
- $r \leftarrow H_3(\mathtt{m})$
- If $\mathtt{c}_1 \neq rP_1$ return $\perp$
- $\mathtt{k} \leftarrow H_5(\mathtt{m})$
- Return $\mathtt{k}$

The security of this construction follows by combining Theorem 2.4 and Theorem 3.4 proved in the next section.

Theorem 3.3. *If H_2, H_3 and H_5 are modelled as random oracles then the above construction is secure against adaptive chosen ciphertext attack. Specifically, if A is a PPT algorithm that breaks the above IBKEM using a chosen ciphertext attack, then there exists a PPT algorithm B, with*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(A) \leq 2eq_2 \cdot (q_3+q_5+q_D)(1+q_X)\left(\mathrm{Adv}^{\mathtt{CBDH}}_{\Gamma}(B)+\frac{1}{q_2\cdot 2^n}\right)+\frac{2q_D}{p}.$$

Here q_2, q_3, q_5, q_D and q_X denote the maximum number of queries that A can place to the H_2, H_3, H_5, decryption and private key extraction oracles respectively.

3.2.3 A Generic Construction of IBKEMs

In [38] Dent proposes a number of generic constructions of KEMs from standard public-key encryption schemes. The KEMs themselves are secure in a strong sense, however the encryption schemes from which they are built require only a weak notion of security. It is this line of work which we aim to extend in this section.

Here we take a generic probabilistic identity-based encryption scheme characterised by $(\mathbb{G}_{\mathtt{IBE}}, \mathbb{X}_{\mathtt{IBE}}, \mathbb{E}_{\mathtt{IBE}}, \mathbb{D}_{\mathtt{IBE}})$. We assume the message space of this scheme is given by $\mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$ and the space of randomness is given by $\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})$. We assume a cryptographic hash function $H_3' : \{0,1\}^* \rightarrow \mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})$.

In the following construction we make no assumption on how such an identity-based scheme is constructed only that it exists. In practice one can take, for instance, the BasicIdent scheme.

$\mathbb{E}_{\mathtt{IBKEM}}(\mathtt{ID}, \mathtt{Mpk})$

- $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{IBE}}(\mathtt{Mpk})$
- $r \leftarrow H_3'(\mathtt{m})$
- $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$
- $\mathtt{k} \leftarrow H_5(\mathtt{m})$
- Return $(\mathtt{k}, \mathtt{c})$

$\mathbb{D}_{\mathtt{IBKEM}}(\mathtt{c}, D)$

- $\mathtt{m} \leftarrow \mathbb{D}_{\mathtt{IBE}}(\mathtt{c}, D)$
- If $\mathtt{m} = \perp$ return $\perp$
- $r \leftarrow H_3'(\mathtt{m})$
- If $\mathtt{c} \neq \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$ return $\perp$
- $\mathtt{k} \leftarrow H_5(\mathtt{m})$
- Return $\mathtt{k}$

This construction is shown to be secure in the next theorem.

Theorem 3.4. *If the underlying identity-based encryption scheme above is OW-CPA secure and H_3' and H_5 are modelled as random oracles then the construction is secure against adaptive chosen ciphertext attack. Specifically, if A is a PPT algorithm that breaks the above IBKEM using a chosen ciphertext attack, then there exists a PPT algorithm B, with*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(A) \leq 2(q_3 + q_5 + q_D) \cdot \mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{IBE}}(B) + \frac{2q_D\gamma(\mathtt{Mpk})}{|\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})|},$$

where q_3, q_5 and q_D are the maximum number of queries made by A to H_3', H_5 and the decryption oracle respectively, and $\gamma(\mathtt{Mpk})$ is as defined in Section 2.3.5.

Proof. Let A denote an IND-CCA adversary against the IBKEM, as specified in the statement of the theorem. Security is proved via a game-hop. In each game $\mathtt{Game}_i$, where $i = 0, 1$, we let S_i denote the event that $b = b'$. We let $\mathtt{Game}_0$ denote the original attack game so that

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBKEM}}(A) = |2\Pr[S_0] - 1|.$$

Let $\mathtt{Game}_1$ be the same as $\mathtt{Game}_0$ except we simulate the H_3', H_5, extraction and

decapsulation queries as described below.

- H_3' **queries:** We maintain a list L_3 which contains at most q_3 pairs $(\mathtt{m}, h_3)$. On input of $\mathtt{m}$, if $(\mathtt{m}, h_3) \in L_3$ then we return h_3, otherwise we select h_3 at random append $(\mathtt{m}, h_3)$ to the list and return h_3.

- H_5 **queries:** We maintain a list L_5 of length at most $q_5 + q_D$ pairs $(\mathtt{m}, h_5)$. On input of $\mathtt{m}$, if $(\mathtt{m}, h_5) \in L_3$ then we return h_5, otherwise we select h_5 at random append $(\mathtt{m}, h_5)$ to the list and return h_5.

- **Extraction queries:** These are answered as in the genuine attack game.

- **Decapsulation queries:** We respond to a query $(\mathtt{c}, \mathtt{ID})$ as follows:

  - If $\mathtt{ID} \neq \mathtt{ID}^*$ then we compute the private key D via the $\mathbb{X}_{\mathtt{IBKEM}}$ oracle and respond as the real decapsulation oracle would.
  - If $\mathtt{ID} = \mathtt{ID}^*$ but $\mathtt{c} \neq \mathtt{c}^*$ we check for each pair $(\mathtt{m}, h_3) \in L_3$, if

    $$\mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; h_3) = \mathtt{c}.$$

    If such a pair exists then run the simulator of H_5 on input $\mathtt{m}$ so as to obtain h_5, we then return h_5. If no such pair exists we return $\perp$.

Note that $\mathtt{Game}_0$ and $\mathtt{Game}_1$ are identical, as A operates in the random oracle model, except if a decapsulation query with input $\mathtt{c}$ where

$$\mathtt{m} = \mathbb{D}_{\mathtt{IBE}}(\mathtt{c}, D) \text{ and } \mathtt{c} = \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; H_3'(\mathtt{m}))$$

is made but no H_3' queries on input $\mathtt{m}$ have been made. Call this event E, hence

$$\Pr[S_0 \wedge \neg E] = \Pr[S_1 \wedge \neg E].$$

From Lemma 2.1 we have

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[E] \leq \frac{q_D \gamma(\mathtt{Mpk})}{|\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})|}.$$

This follows from the fact that, for each decryption query for which $\mathtt{ID} = \mathtt{ID}^*$ and $\mathtt{c} \neq \mathtt{c}^*$ but for no pair $(\mathtt{m}, h_3) \in L_3$ do we have $\mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; h_3) = \mathtt{c}$, there is probability at most $\gamma(\mathtt{Mpk})/|\mathcal{R}_{\mathtt{IBE}}(\mathtt{Mpk})|$ that for r chosen at random there is an $\mathtt{m}$ such that $\mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r) = \mathtt{c}$; moreover, there are at most q_D such queries.

Let $\mathtt{m}^*$ denote the hidden value encrypted by $\mathtt{c}^*$. We let E' denote the event that the attacker queries either H_3' or H_5 with $\mathtt{m}^*$. Then, since we are in the random oracle model,

$$\Pr[S_1] = \Pr[S_1 \wedge E'] + \Pr[S_2 \wedge \neg E'] = \Pr[S_1 \wedge E'] + \frac{1}{2}.$$

Hence $|\Pr[S_1] - 1/2| \leq \Pr[S_1|E']$. The theorem follows if we can describe an algorithm B such that

$$\mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{IBE}}(B) = \Pr[S_1|E']/(q_3 + q_5 + q_D).$$

Algorithm B operates as follows. On input of $\mathtt{Mpk}$ it simply relays this onto algorithm A's first stage. Algorithm A then responds with an identity $\mathtt{ID}^*$. This value of $\mathtt{ID}^*$ is passed onto B's challenger who responds with an encryption $\mathtt{c}^*$ of a random message $\mathtt{m}^*$. The value of $\mathtt{c}^*$ is then passed onto A's second stage, along with a random key $\mathtt{k}^*$ from the co-domain of H_5. Note that if H_3' is queried on $\mathtt{m}^*$ then A can see that the encapsulation $\mathtt{c}^*$ has not been computed correctly. All oracle queries are answered as in $\mathtt{Game}_1$, except the private key extraction queries which are now answered using the oracle given to algorithm B. At the end of the game algorithm B picks a random value which has been input into H_3' or H_5 from

the L_3 or L_5 list and returns this as its guess for $\mathtt{m}^*$. It is clear that $\mathrm{Adv}^{\mathtt{OW-CPA}}_{\mathtt{IBE}}(B) = \Pr[S_1|E']/(q_3+q_5+q_D)$ and so the result follows. □

We note that Theorem 3.4 was used by Chen et al. in [31] to produce an IBKEM based on the IBE scheme of Sakai and Kasahara (see Section 2.4.6). When combined with a DEM, this gives the most efficient IBE scheme currently known.

3.2.4 Comparison with FullIdent

In the following table we count the various operation counts for encryption and decryption of our identity-based encryption schemes. Note we make no distinction as to in which group exponentiations occur since one can select the group so as to make the operation more efficient in any given implementation. We let C-1, denote Lynn's KEM construction in Section 3.2.1 combined with a DEM, and C-2 denote our second construction in Section 3.2.2 combined with a DEM, which is an optimised version of our third generic construction in Section 3.2.3.

	Pairings		Exp's		Hash Fncs		Message
Scheme	$\mathbb{E}_{\mathtt{IBE}}$	$\mathbb{D}_{\mathtt{IBE}}$	$\mathbb{E}_{\mathtt{IBE}}$	$\mathbb{D}_{\mathtt{IBE}}$	$\mathbb{E}_{\mathtt{IBE}}$	$\mathbb{D}_{\mathtt{IBE}}$	Size
FullIdent-1	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert\mathtt{m}\rvert$
FullIdent-2	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert\mathbb{E}_{\mathtt{SKE}}(\mathtt{m})\rvert$
C-1	1	1	2	0	2	1	$\lvert G_1\rvert + \lvert\mathbb{E}_{\mathtt{DEM}}(\mathtt{m})\rvert$
C-2	1	1	2	1	4	3	$\lvert G_1\rvert + n + \lvert\mathbb{E}_{\mathtt{DEM}}(\mathtt{m})\rvert$

From the table we see that the KEM-DEM approach with Lynn's method is marginally more efficient than the technique of Boneh and Franklin, or the IBKEM approach of our construction two. We also note that FullIdent-2 requires a less stringent definition of security of the symmetric encryption scheme, this is because chosen-ciphertext security is provided by the Fujisaki–Okamoto transformation rather than the CCA security of the symmetric cipher.

We also need to compare the tightness of the security guarantees offered by the various constructions. To do this we make an explicit numerical comparison. We take a security parameter where for our bilinear groups which results in a value of $p \approx 2^{\lambda}$. We assume adversaries against our schemes exists which make at most $q_X \approx 2^{32}$ calls to their key extraction oracles and at most $q_D \approx 2^{32}$ calls to their decryption oracles. For our GBDH adversary we also limit the number of DBDH queries to 2^{32}. We also assume that the number of hash function queries is bounded, for each hash function, by approximately 2^{32}. We then obtain the following tightness results:

FullIdent-1 :

$$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-CCA}}(A) \leq 2^{99} \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(B) + 2^{67-n}.$$

FullIdent-2 :

$$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-CCA}}(A) \leq 2^{99} \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(B) + 2^{67-n} + 2^{34}\mathrm{Adv}_{\mathtt{SKE}}^{\mathtt{FG-CPA}}(C).$$

C-1 :

$$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-CCA}}(A) \leq 2^{34} \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B, 2^{32}) + 2^{33-\lambda} + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG-CCA}}(C).$$

C-2 :

$$\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{IND-CCA}}(A) \leq 2^{100}\mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(B) + 2^{67-n} + 2^{33-\lambda} + \mathrm{Adv}_{\mathtt{DEM}}^{\mathtt{FG-CCA}}(C).$$

Suppose we use an elliptic curve system to implement our bilinear groups, with MOV embedding degree d (see [69] for a definition). We then know that there exists a sub-exponential algorithm to solve the CBDH and GBDH algorithms

with running time essentially [78]

$$L_{2^{d\lambda}}(1/3,(64/9)^{1/3}) = \exp\left((64/9)^{1/3}(\log 2^{d\lambda})^{1/3}(\log\log 2^{d\lambda})^{2/3}\right).$$

In addition there is an exponential algorithm [73] with running time essentially

$$2^{\lambda/2}.$$

Hence, we can estimate a lower bound for $\mathrm{Adv}_{\Gamma}^{\mathtt{CBDH}}(B)$ and $\mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B,c)$ via

$$1/\min(L_{2^{d\lambda}}(1/3,(64/9)^{1/3}), 2^{\lambda/2}).$$

Suppose we wished to obtain a security guarantee of an advantage of the adversary A against our identity-based scheme of 2^{-80}, in addition we assume that our symmetric cipher is chosen so that it can guarantee an advantage of $2^{-\mu}$. We now look at what this implies for the parameters of the pairing based parts of the scheme.

FullIdent-1 :

$$n \geq 147 \text{ and } \lambda \geq \max(358, 5700/d) \approx 950 \text{ (if } d = 6 \text{).}$$

FullIdent-2 :

$$n \geq 147, \;\; \mu \geq 114 \text{ and } \lambda \geq \max(358, 5700/d) \approx 950 \text{ (if } d = 6 \text{).}$$

C-1 :

$$\mu \geq 80 \text{ and } \lambda \geq \max(112, 226, 1900/d) \approx 316 \text{ (if } d = 6 \text{).}$$

C-2 :

$$n \geq 147, \;\; \mu \geq 80 \text{ and } \lambda \geq \max(113, 360, 5740/d) \approx 957 \text{ (if } d = 6 \text{).}$$

Notice, the weaker security requirement on the symmetric cipher in FullIdent-2, is not such an advantage when one considers the tightness result. Also note, the tightness of the reduction of our first construction, needs to be balanced against the fact that one is reducing to a possibly easier problem.

Chapter 4

Multi-Recipient Schemes

In this chapter we first recall the definition of a secure multi-recipient public-key encryption scheme as introduced in [14]. We then consider a natural extension to key encapsulation mechanisms, which are closely related to the mKEM notion proposed by Smart in [83]. Next we consider IBKEMs, as introduced in Chapter 3, in the multi-recipient settings and present an efficient multi-recipient IBKEM. We prove our scheme secure and conclude with an analysis of its efficiency. The work in this chapter first appeared in [9, 10] and represents joint research with Manuel Barbosa.

4.1 Definitions

4.1.1 Multi-Recipient PKE Schemes

An m-Multi-Recipient PKE (m-MR-PKE) scheme [14] is defined similarly to a standard PKE scheme, with two exceptions: (1) the key generation algorithm is parameterised with a domain parameter I which ensures compatibility between users' key pairs and various spaces[1]; and (2) the encryption algorithm takes a list

[1]This parameter is generated once for all users and henceforth, unless specifically stated otherwise, we leave its generation as implicit to simplify notation. See the definition of PKE schemes in

of m message/public key pairs and outputs a list of m ciphertexts. More formally, an m-MR-PKE consists of the following algorithms:

- $\mathbb{G}_{m-\mathtt{MR}-\mathtt{PKE}}(I)$: This is the probabilistic key generation algorithm which takes as input the domain parameter I and returns a secret key $\mathtt{SK}$ and a matching public key $\mathtt{PK}$.

- $\mathbb{E}_{m-\mathtt{MR}-\mathtt{PKE}}((\mathtt{m}_i, \mathtt{PK}_i)_{i=1}^{m}; r)$: This is the probabilistic encryption algorithm, which on input messages $\mathtt{m}_i \in \mathcal{M}_{m-\mathtt{MR}-\mathtt{PKE}}(\mathtt{PK}_i)$, public keys $\mathtt{PK}_i$ and possibly some randomness $r \in \mathcal{R}_{m-\mathtt{MR}-\mathtt{PKE}}(I)$ outputs a ciphertext vector $(\mathtt{c}_1, \ldots, \mathtt{c}_m)$ with $\mathtt{c}_i \in \mathcal{C}_{m-\mathtt{MR}-\mathtt{PKE}}(\mathtt{PK}_i)$.

- $\mathbb{D}_{m-\mathtt{MR}-\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$: This is the deterministic decryption algorithm. On input a ciphertext $\mathtt{c}$ and a private key $\mathtt{SK}$ this algorithm outputs a message $\mathtt{m} \in \mathcal{M}_{m-\mathtt{MR}-\mathtt{PKE}}(\mathtt{PK})$ or a special failure symbol $\perp$.

The correctness of an m-MR-PKE is defined in the obvious way that each ciphertext component should decrypt properly under the appropriate secret key.

Given a PKE scheme we can build the *associated* m-MR-PKE scheme as follows. The key generation and decryption algorithms are identical to the underlying PKE scheme (which we call the *base* scheme). Encryption is defined naturally by running multiple parallel instances of the base encryption algorithm. In case there are common parameters that may be shared by all public keys to improve overall efficiency (for example in an ElGamal-based scheme, this might include a domain modulus and a generator which all parties use to create key pairs), these are included in the domain parameter I. The formal security model for an m-MR-PKE, as defined in [14], considers the possibility of insider attacks by allowing the adversary to corrupt some of the users by maliciously choosing their public and private keys. This ensures that security is still in place between the legitimate recipients,

Section 2.3.2.

or in other words, that there is no "cross-talk" between the ciphertexts intended for different recipients.

IND-atk

1. $t \leftarrow A_0(I)$ with $1 \leq t \leq m$
2. For $i = 1, \ldots, t$
   $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{m-\mathtt{MR}-\mathtt{PKE}}(I)$
3. $((\mathtt{m}_{i,0}, \mathtt{m}_{i,1})_{i=1}^{t}, (\mathtt{SK}_i, \mathtt{PK}_i, \mathtt{m}_i)_{i=t+1}^{m}, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{PK}_1, \ldots, \mathtt{PK}_t)$
4. $b \leftarrow \{0,1\}$
5. $\mathbf{c}^* \leftarrow \mathbb{E}_{m-\mathtt{MR}-\mathtt{PKE}}((\mathtt{m}_{i,b}, \mathtt{PK}_i)_{i=1}^{t}, (\mathtt{m}_i, \mathtt{PK}_i)_{i=t+1}^{m})$
6. $b' \leftarrow A_2^{O_2}(\mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{atk}}_{m-\mathtt{MR}-\mathtt{PKE}}(A) := |2\Pr[b' = b] - 1|.$$

The adversary runs in three stages: (1) given the global parameters, it chooses the number of non-corrupt users t in the game; (2) given the public keys for the legitimate users, it produces all the information that is necessary to build a complete challenge (this includes pairs of plaintext messages for the legitimate users, single plaintext messages for corrupt users and the corresponding key pairs[2]); and (3) given the challenge, it guesses the value of the hidden bit. Throughout the game, the adversary also has access to O_1 and O_2, which denote a set of (possibly null) oracles:

- If $\mathtt{atk} = \mathtt{CPA}$ then $O_1 = O_2 = \text{null}$.
- If $\mathtt{atk} = \mathtt{CCA}$ then O_1 is a set of decryption oracles one for each $\mathtt{PK}_i$ and O_2 is

[2]The fact that the adversary provides the secret keys of corrupt users is justified by observing that access to decryptions under public keys of the adversary's choice is an unrealistic attack scenario (c.f. the last remark in Section 2.3.6).

same as O_1 except that no component c_i^* of the challenge ciphertext $\mathbf{c}^*$ could be submitted to an oracle corresponding to $\mathtt{PK}_i$.

Remark. The above security model could be strengthened by giving the adversary access to a left-right oracle and the ability to adaptively corrupt users as follows: (1) an honest user generation oracle where public keys are provided to the adversary; (2) a malicious user generation where the adversary provides maliciously chosen public keys; and (3) a user corruption oracle, where the adversary can ask for the secret key of an honest user. The left-right oracle should be restricted so that it can only be called with distinct messages on non-corrupt and non-malicious users. We do not pursue this model here further and note that one might be able to establish its equivalence to the single-user security model via a hybrid argument.

Remark. Note also that m is not an input to any of the algorithms: it is a parameter of the definition of the primitive. One could consider an alternative MR-PKE primitive where m is taken as an input by the encryption algorithm. In this case, it is not necessarily the case that if an MR-PKE is secure for m users then it is also secure for $m-1$ users if m is *chosen by the challenger* in the security definition. For a uniform treatment we should let m to be chosen in an adversarial way. Since all the schemes studied in this thesis are built out of a base PKE scheme (and the proofs hold for any m polynomial in the security parameter), their associated MR-PKE schemes satisfy this uniform security property automatically. We therefore adopt the standard definition as given in [14].

In this work we are interested in a special case of m-MR-PKE schemes where the same message is sent to all recipients. We refer to this special case as a *single-message* multi-recipient scheme (m-SM-PKE in short). This specific case is a recurring use-case of m-MR-PKEs in practice.

- $\mathbb{G}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$: This is the probabilistic key generation algorithm which takes

as input the domain parameter I and returns a secret key $\mathtt{SK}$ and a matching public key $\mathtt{PK}$.

- $\mathbb{E}_{m-\mathtt{SM}-\mathtt{PKE}}(\mathtt{m}(\mathtt{PK}_i)_{i=1}^m; r)$: This is the probabilistic encryption algorithm. On input a messages $\mathtt{m} \in \mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$, public keys $\mathtt{PK}_i$ and possibly some randomness $r \in \mathcal{R}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$ this outputs a ciphertext vector $(\mathtt{c}_1, \ldots, \mathtt{c}_m)$ with $\mathtt{c}_i \in \mathcal{C}_{m-\mathtt{SM}-\mathtt{PKE}}(\mathtt{PK}_i)$.

- $\mathbb{D}_{m-\mathtt{SM}-\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$: This is the deterministic decryption algorithm. On input a ciphertext $\mathtt{c}$ and a private key $\mathtt{SK}$ this algorithm outputs a message $\mathtt{m} \in \mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$ or a special failure symbol $\perp$.

Below is the simplified security model for m-SM-PKEs.

IND-atk

1. For $i = 1, \ldots, m$
   $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$
3. $b \leftarrow \{0, 1\}$
4. $\mathbf{c}^* \leftarrow \mathbb{E}_{m-\mathtt{SM}-\mathtt{PKE}}(\mathtt{m}_b, (\mathtt{PK}_i)_{i=1}^m)$
5. $b' \leftarrow A_2^{O_2}(\mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{atk}}_{m-\mathtt{SM}-\mathtt{PKE}}(A) := |2\Pr[b' = b] - 1|.$$

There is an important difference to the m-MR-PKE model: the adversary is no longer able to corrupt users. The reason for this is that, since all recipients will be getting the same message, there is no need to enforce security across the individual ciphertexts[3]. The oracles to which the adversary has access are as in the multi-

[3]The security definition for an m-MR-PKE when considered in the single-message setting gives rise to the single- message security definition. This follows since in this case we must have $\mathtt{m}_{i,b} =$

message game. We will also see in the next section that this model is particularly relevant in the hybrid encryption scenario.

4.1.2 Multi-Recipient KEMs

An *m*-Multi-Recipient KEM (*m*-MR-KEM) is identical to a standard KEM, except that the encapsulation algorithm takes a list of *m* public keys and outputs a list of *m* ciphertext/key pairs.

- $\mathbb{G}_{m-\mathtt{MR}-\mathtt{KEM}}(I)$: This is the probabilistic key generation algorithm which takes as input the domain parameter I and returns a secret key $\mathtt{SK}$ and a matching public key $\mathtt{PK}$.

- $\mathbb{E}_{m-\mathtt{MR}-\mathtt{KEM}}((\mathtt{PK}_i)_{i=1}^m)$: This is the probabilistic encapsulation algorithm. On input public keys $\mathtt{PK}_i$, this outputs an encapsulation vector $(\mathtt{k}_i\mathtt{c}_i)_{i=1}^m$ where $\mathtt{c}_i \in \mathcal{C}_{m-\mathtt{MR}-\mathtt{KEM}}(\mathtt{PK}_i)$ is an encapsulation of $\mathtt{k}_i$ under $\mathtt{PK}_i$.

- $\mathbb{D}_{m-\mathtt{MR}-\mathtt{KEM}}(\mathtt{c},\mathtt{SK})$: This is the deterministic decapsulation algorithm. On input a ciphertext $\mathtt{c}$ and a private key $\mathtt{SK}$ this algorithm outputs a key $\mathtt{k} \in \mathcal{K}_{m-\mathtt{MR}-\mathtt{KEM}}(\mathtt{PK})$ or a special failure symbol $\perp$.

The correctness of an *m*-MR-KEM is defined by requiring each encapsulation component to decapsulate properly under the appropriate secret key.

Again, given a KEM we can build the *associated m*-MR-KEM as follows. The key generation and decapsulation algorithms are identical to the underlying KEM (which we call the *base* KEM). Encapsulation is defined naturally by using multiple parallel instances of the base KEM algorithm.

A direct adaptation to *m*-MR-KEMs of the full multiple-recipient PKE security model, described in the previous section, is shown below. We maintain the possi-

$\mathtt{m}_{j,b} = \mathtt{m}_b$ and $\mathtt{m}_i = \mathtt{m}_b$, which would mean either all messages are the same or there are no corrupted users.

bility of inside attacks since we want to ensure legitimate users cannot eavesdrop on each others' ciphertexts.

IND-atk

1. $t \leftarrow A_0(I)$ with $1 \leq t \leq m$
2. For $i = 1, \ldots, t$
   $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{m-\mathtt{MR}-\mathtt{KEM}}(I)$
3. $((\mathtt{SK}_i, \mathtt{PK}_i)_{i=t+1}^{m}, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{PK}_1, \ldots, \mathtt{PK}_t)$
4. $(\mathtt{k}_{i,0})_{i=1}^{t} \leftarrow \mathcal{K}_{m-\mathtt{MR}-\mathtt{KEM}}(I)$
5. $((\mathtt{k}_{i,1}, \mathtt{c}_i^*)_{i=1}^{t}, (\mathtt{k}_i, \mathtt{c}_i^*)_{i=t+1}^{m}) \leftarrow \mathbb{E}_{m-\mathtt{MR}-\mathtt{KEM}}((\mathtt{PK}_i)_{i=1}^{m})$
6. $b \leftarrow \{0,1\}$
7. $b' \leftarrow A_2^{O_2}((\mathtt{k}_{i,b}, \mathtt{c}_i^*)_{i=1}^{t}, (\mathtt{k}_i, \mathtt{c}_i^*)_{i=t+1}^{m}, \mathrm{st})$

$$\mathrm{Adv}_{m-\mathtt{MR}-\mathtt{KEM}}^{\mathtt{IND}-\mathtt{atk}}(A) := |2\Pr[b' = b] - 1|.$$

In the above, O_1 and O_2 denote a set of oracles:

- If $\mathtt{atk} = \mathtt{CPA}$ then $O_1 = O_2 = \text{null}$.
- If $\mathtt{atk} = \mathtt{CCA}$ then O_1 is a set of decapsulation oracles one for each $\mathtt{PK}_i$ and O_2 is same as O_1 except that no component $\mathtt{c}_i^*$ of the challenge encapsulation $\mathbf{c}^* = (\mathtt{c}_i^*)_{i=1}^{m}$ could be submitted to an oracle corresponding to $\mathtt{PK}_i$.

Smart [83] defines the mKEM primitive, a key encapsulation mechanism which translates the hybrid encryption paradigm to the multi-cast setting: it permits encapsulating the same session key to several different receivers. The point is that encrypting a single message to all these recipients can then be done using a single DEM instantiation based on that unique key rather than m different ones. This provides, not only reduced processing load, but also bandwidth savings, and captures a common use of hybrid encryption in practice. The mKEM primitive is as follows.

- $\mathbb{G}_{\mathtt{mKEM}}(1^\kappa)$ which is a probabilistic key generation algorithm. On input of 1^κ and the domain parameters, this algorithm outputs a public/private key pair $(\mathtt{PK}, \mathtt{SK})$.

- $\mathbb{E}_{\mathtt{mKEM}}(\mathcal{PK})$ which is a probabilistic encapsulation algorithm. On input a set of public keys $\mathcal{PK} = \{\mathtt{PK}_1, \ldots, \mathtt{PK}_m\}$ this algorithm outputs an encapsulated key-pair $(\mathtt{k}, \mathbf{c})$, where $\mathtt{k} \in \mathcal{K}_{\mathtt{mKEM}}(\kappa)$ is the session key and $\mathbf{c}$ is an encapsulation of the key $\mathtt{k}$ under the public keys $\{\mathtt{PK}_1, \ldots, \mathtt{PK}_m\}$.

- $\mathbb{D}_{\mathtt{mKEM}}(\mathbf{c}, \mathtt{SK})$ which is a decapsulation algorithm. This takes as input an encapsulation $\mathbf{c}$ and a private key $\mathtt{SK}$, and outputs a key $\mathtt{k}$ or $\perp$.

An mKEM is *correct* if the following experiment returns 1 with probability 1.

1. $(\mathtt{PK}_i, \mathtt{SK}_i) \leftarrow \mathbb{G}_{\mathtt{mKEM}}(1^\kappa)$ for $1 \leq i \leq m$
2. $(\mathtt{k}, \mathbf{c}) \leftarrow \mathbb{E}_{\mathtt{mKEM}}(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$
3. $j \leftarrow \{1, \ldots, m\}$
4. If $\mathtt{k} = \mathbb{D}_{\mathtt{mKEM}}(\mathbf{c}, \mathtt{SK}_j)$ return 1 else return 0

Security of an mKEM is defined via the following game.

(m,n)-IND-gCCA

1. $(\mathtt{PK}_i, \mathtt{SK}_i) \leftarrow \mathbb{G}_{\mathtt{mKEM}}(1^\kappa)$ for $i = 1, \ldots, n$
2. $\mathcal{PK} \leftarrow \{\{\mathtt{PK}_1, \ldots, \mathtt{PK}_n\}\}$
3. $(\mathcal{PK}^*, \mathrm{st}) \leftarrow A_1^{O_1}(\mathcal{PK})$
4. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{mKEM}}(\kappa)$
5. $(\mathtt{k}_1, \mathbf{c}^*) \leftarrow \mathbb{E}_{\mathtt{mKEM}}(\mathcal{PK}^*)$
6. $b \leftarrow \{0,1\}$
7. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b, \mathbf{c}^*, \mathrm{st})$

The advantage is defined in the usual way. In step 3 above, $\mathcal{PK}^* \subseteq \mathcal{PK}$ and $m = |\mathcal{PK}^*| \leq n$. Here O_i denote the decapsulation oracles to which the adversary has access in rounds 1 and 2. As it is noted in [83], restricting access to this oracle by simply disallowing the query $(\mathbf{c}^*, \mathtt{PK}^*)$ for some $\mathtt{PK}^* \in \mathcal{PK}^*$ would be too lenient. Consider an mKEM in which an encapsulation is of the form $\mathbf{c}^* = \mathtt{c}_1^* || \ldots || \mathtt{c}_m^*$ where each $\mathtt{c}_i^*$ is intended for the recipient with public key $\mathtt{PK}_i^*$. On reception of a challenge encapsulation of this form, the adversary could query the decapsulation oracle with a pair $(\mathtt{c}_i^*, \mathtt{PK}_i^*)$, which is *not* disallowed, and win the game.

To avoid this type of problem, Smart [83] defines a more restrictive, but still reasonable, oracle access. The restriction imposed in the model is that in the second stage the adversary is only allowed to query the decapsulation oracle with those $\mathbf{c}$ which decapsulate to a different key from that encapsulated by $\mathbf{c}^*$. This means the ciphertexts could be benignly malleable (see Section 2.3.2).

The primitive and security definitions we adopt for such single-key KEMs are slightly different from the mKEM primitive. In this alternative definition, referred to as an m-SK-KEM, the encapsulation algorithm takes a list of m public keys and outputs an encapsulation for each user. The decapsulation algorithm takes the *designated* ciphertext for a user.

- $\mathbb{G}_{m-\mathtt{SK}-\mathtt{KEM}}(I)$: This is the probabilistic key generation algorithm which takes as input the domain parameter I and returns a secret key $\mathtt{SK}$ and a matching public key $\mathtt{PK}$.

- $\mathbb{E}_{m-\mathtt{SK}-\mathtt{KEM}}((\mathtt{PK}_i)_{i=1}^m)$: This is the probabilistic encapsulation algorithm. On input public keys $\mathtt{PK}_i$, this outputs an encapsulation vector $(\mathtt{k}, (\mathtt{c}_i)_{i=1}^m$ where $\mathtt{c}_i \in \mathcal{C}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{PK}_i)$ is an encapsulation of $\mathtt{k}$ under $\mathtt{PK}_i$.

- $\mathbb{D}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{c}, \mathtt{SK})$: This is the deterministic decapsulation algorithm. On input a ciphertext $\mathtt{c}$ and a private key $\mathtt{SK}$ this algorithm outputs a key $\mathtt{k} \in$

$\mathcal{K}_{m-\mathtt{SK}-\mathtt{KEM}}(I)$ or a special failure symbol $\perp$.

We believe our choice of primitive captures more closely multi-cast applications where each user decapsulates his share of the ciphertext. Note that since a KEM assigns independent encapsulated keys to each public key, one cannot build an m-SK-KEM (or an mKEM) by running the KEM's encapsulation algorithm multiple times. Let us now present the security model for m-SK-KEMs and compare it with that of mKEMs.

IND-atk

1. For $i = 1, \ldots, m$
   $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{m-\mathtt{SK}-\mathtt{KEM}}(I)$
2. $\mathrm{st} \leftarrow A_1^{\mathcal{O}_1}(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{m-\mathtt{SK}-\mathtt{KEM}}(I)$
4. $(\mathtt{k}_1, (\mathtt{c}_i^*)_{i=1}^m) \leftarrow \mathbb{E}_{m-\mathtt{SK}-\mathtt{KEM}}((\mathtt{PK}_i)_{i=1}^m)$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{k}_b, (\mathtt{c}_i^*)_{i=1}^m, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{atk}}_{m-\mathtt{SK}-\mathtt{KEM}}(A) := |2\Pr[b' = b] - 1|.$$

The oracles to which the adversary has access are as in the multi-key game.

To compare the security models, note that one can naturally build an mKEM out of an m-SK-KEM. Conversely, an m-SK-KEM could be built using an mKEM by setting each user's ciphertext to be that returned by the mKEM's encapsulation algorithm. One then can easily show that the associated mKEM of a secure m-SK-KEM is also secure. However, the converse does not hold if the underlying mKEM is benignly malleable. This comparison is, however, somewhat artificial since for concrete schemes one can build a better m-SK-KEM by letting a user's ciphertext

to consist of only those components of the encapsulation which are essential to that particular user.

4.1.3 Multi-Recipient IBE Schemes

The IBE primitive can be extended to the multi-recipient setting in an analogous manner: the encryption algorithm takes a list of m message/identity pairs and outputs a list of m ciphertexts. For m-MR-IBEs the master public key acts as the domain parameter I. One can restrict this notion to the single-message setting and build m-SM-IBEs. We give the details of the security model for m-MR-IBEs for completeness.

IND-atk

1. $(\mathtt{Mpk}, \mathtt{Msk}) \leftarrow \mathbb{G}_{m-\mathtt{MR}-\mathtt{IBE}}(1^\kappa)$
2. $((\mathtt{m}_{i,0}, \mathtt{m}_{i,1}, \mathtt{ID}_i^*)_{i=1}^t, (\mathtt{m}_i, \mathtt{ID}_i^*)_{i=t+1}^m, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk})$
3. $b \leftarrow \{0,1\}$
4. $\mathbf{c}^* \leftarrow \mathbb{E}_{m-\mathtt{MR}-\mathtt{IBE}}((\mathtt{m}_{i,b}, \mathtt{ID}_i)_{i=1}^t, (\mathtt{m}_i, \mathtt{ID}_i)_{i=t+1}^m, \mathtt{Mpk})$
5. $b' \leftarrow A_2^{O_2}(\mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{atk}}_{m-\mathtt{MR}-\mathtt{IBE}}(A) := |2\Pr[b' = b] - 1|.$$

In the above, O_1 and O_2 denote a set of oracles as follows:

- If $\mathtt{atk} = \mathtt{CPA}$ then O_1 and O_2 denote key extraction oracle with the restriction they cannot be queries on $\mathtt{ID}_1^*, \ldots, \mathtt{ID}_t^*$.
- If $\mathtt{atk} = \mathtt{CCA}$ then O_1 and O_2 are as above together with a set of decapsulation oracles one for each $\mathtt{ID}_i$ with the usual restriction that no component $\mathtt{c}_i^*$ of the challenge ciphertext $\mathbf{c}^*$ can be submitted to an oracle in O_2 corresponding to $\mathtt{ID}_i$.

Note that we no longer need to specify the corrupt users due to the presence of the extraction oracle. The security model for *m*-SM-IBEs can be easily derived from the above game.

4.1.4 Multi-Recipient IBKEMs

In this section we define what we mean by an identity-based key encapsulation to multiple parties or an *m*-SK-IBKEM. This primitive is an adaptation of the *m*-SK-KEM primitive (as opposed to the mKEM primitive) to the identity-based setting. An *m*-SK-IBKEM is a four-tuple of PPT algorithms as follows.

- $\mathbb{G}_{m-\mathtt{SK}-\mathtt{IBKEM}}(1^\kappa)$. A probabilistic algorithm which takes as input a security parameter 1^κ and returns the master public key $\mathtt{Mpk}$ and the master secret key $\mathtt{Msk}$. We assume, as usual, that $\mathtt{Mpk}$ contains all system parameters, including the security parameter κ and is available as the domain parameter to all users.

- $\mathbb{X}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID},\mathtt{Msk})$. A (possibly) probabilistic algorithm which takes as input an identifier string $\mathtt{ID} \in \{0,1\}^*$ and $\mathtt{Msk}$, and returns the associated private key D.

- $\mathbb{E}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID}_1,\ldots,\mathtt{ID}_m,\mathtt{Mpk})$. This is the probabilistic encapsulation algorithm. On input a tuple of identities and the master public key, this outputs a pair $(\mathtt{k},\mathbf{c})$, where $\mathtt{k}$ is a session key in $\mathcal{K}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{Mpk})$ and $\mathbf{c} = (\mathtt{c}_1,\ldots,\mathtt{c}_m)$, where each $\mathtt{c}_i \in \mathcal{C}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{Mpk})$ is an encapsulation of $\mathtt{k}$ for user $\mathtt{ID}_i$.

- $\mathbb{D}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{c},D)$. This is the deterministic decapsulation algorithm. On input an encapsulation $\mathtt{c}$ and a private key D this outputs a key $\mathtt{k}$ or a failure symbol $\perp$.

An m-SK-IBKEM is called *correct* if for all m-tuple of identities $(\mathtt{ID}_1, \ldots, \mathtt{ID}_m)$ the following experiment returns 1 with probability 1.

1. $(\mathtt{Mpk}, \mathtt{Msk}) \leftarrow \mathbb{G}_{m-\mathtt{SK}-\mathtt{IBKEM}}(1^\kappa)$
2. $(\mathtt{k}, \mathtt{c}_1, \ldots, \mathtt{c}_m) \leftarrow \mathbb{E}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID}_1, \ldots, \mathtt{ID}_m, \mathtt{Mpk})$
3. $D_i \leftarrow \mathbb{X}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID}_i, \mathtt{Msk}), 1 \leq i \leq m$
4. $j \leftarrow \{1, \ldots, m\}$
5. If $\mathtt{k} = \mathbb{D}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{c}_j, D_j)$ return 1 else return 0

We define the security of an m-SK-IBKEM according to the following two-stage game.

IND-CCA

1. $(\mathtt{Mpk}, \mathtt{Msk}) \leftarrow \mathbb{G}_{m-\mathtt{SK}-\mathtt{IBKEM}}(1^\kappa)$
2. $(\mathtt{ID}_1^*, \ldots, \mathtt{ID}_m^*, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk})$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{Mpk})$
4. $(\mathtt{k}_1, (\mathtt{c}_i^*)_{i=1}^m) \leftarrow \mathbb{E}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID}_1^*, \ldots, \mathtt{ID}_m^*, \mathtt{Mpk})$
5. $b \leftarrow \{0, 1\}$
6. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b, (\mathtt{c}_i^*)_{i=1}^m, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{CCA}}_{m-\mathtt{SK}-\mathtt{IBKEM}}(A) := |2\Pr[b' = b] - 1|.$$

In the above O_i denotes oracles to which the adversary has access. The adversary has access to a private key extraction oracle which, on input of ID, will output the corresponding value of D. The extracted identity should not be one of the challenge identities. The adversary can also query a decapsulation oracle with respect to any identity ID of its choice. It has access to this decapsulation oracle, subject to the restriction that in the second phase A is not allowed to call it with any pair

$(\mathtt{c}_i^*, \mathtt{ID}_i^*)$, where we assume the adversary is challenged on $\mathbf{c}^* = (\mathtt{c}_1^*, \ldots, \mathtt{c}_m^*)$[4].

4.2 The Composition Theorem

We now prove our multi-recipient IBKEM-DEM composition theorem which allows for a secure multi-recipient IBE to be built. We prove this theorem for single-key IBKEMs and single-message IBEs, as these are primarily studied in this work. One could prove in almost an identical manner a composition theorem for multi-key variants as well as the standard *m*-SK-KEMs and *m*-MR-KEMs. To avoid repetition we omit the details. Note also that this theorem implies the promised single-user IBKEM-DEM composition theorem in Section 3.1.2.

Theorem 4.1. *If an m-SK-IBKEM is IND-CCA secure and a DEM is FG-CCA secure, then the resulting hybrid m-SM-IBE scheme is secure in the IND-CCA sense. More precisely, for any PPT adversary A against the hybrid scheme there are PPT adversaries B_1 and B_2 against the m-SK-IBKEM and the DEM such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SM-IBE}}(A) \leq 2\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK-IBKEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2).$$

Proof. Our proof strategy is as follows. We define a sequence $\mathtt{Game}_0, \mathtt{Game}_1$ and $\mathtt{Game}_2$ of modified attack games in which A runs. The only difference between games is how the environment responds to A's oracle queries.

We fix some notation that we will use throughout. Let $\mathbf{c}^* = (\mathtt{c}_{i,1}^*, \mathtt{c}_{i,2}^*)_{i=1}^m$ be the challenge ciphertext presented to A by its challenge encryption oracle, the oracle that encrypts either $\mathtt{m}_0$ or $\mathtt{m}_1$ according to a bit b. Let $\mathtt{k}^*$ denote the symmetric key used by the challenge encryption oracle in the generation of the challenge ciphertext, or alternatively, the decapsulation of $\mathtt{c}_i^*$ using the secret keys associated

[4]Note that this accounts for cases where the adversary includes repetitions in the challenge identities.

to $\mathtt{ID}_i^*$, the identities chosen by the adversary on which it wishes to be challenged. We let S_i be the event that $b' = b$ in game $\mathtt{Game}_i$ where $i = 0, 1, 2$. Here b is the bit chosen by A's challenge encryption oracle.

Let $\mathtt{Game}_0$ be the genuine attack game played by A. So by definition we have

$$|\Pr[S_0] - 1/2| = \frac{1}{2}\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK-IBKEM}}(A).$$

Game $\mathtt{Game}_0$ is now modified so that whenever an identity $\mathtt{ID}$ and $(\mathtt{c}_1, \mathtt{c}_2)$ is presented to the decryption oracle after the invocation of the challenge encryption oracle, if $\mathtt{ID} = \mathtt{ID}_i^*$ and $\mathtt{c}_1 = \mathtt{c}_{i,1}^*$ then the decryption oracle does not use the genuine decryption procedure for the hybrid scheme, instead it uses the key $\mathtt{k}^*$ to decapsulate $\mathtt{c}_2$ and returns the result to the adversary. This modification to $\mathtt{Game}_0$ gives us the game $\mathtt{Game}_1$. Games $\mathtt{Game}_0$ and $\mathtt{Game}_1$ are identical under the correctness condition of the m-SK-IBKEM, and so $\Pr[S_1] = \Pr[S_0]$.

We now modify $\mathtt{Game}_1$ by replacing $\mathtt{k}^*$ with a random key $\mathtt{k}'$ from $\mathcal{K}_{\mathtt{DEM}}(\kappa, \kappa')$, where κ and κ' are the security parameters of the m-SK-IBKEM and DEM respectively. With this modification we have the game $\mathtt{Game}_2$. The result then follows from the following two lemmas using the triangle inequality. □

Lemma 4.1. *There is a PPT algorithm B_1, whose running time is essentially the same as that of A, such that*

$$|\Pr[S_2] - \Pr[S_1]| = \mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SM-IBE}}(B_1).$$

Proof. To prove this we demonstrate how to construct an adversary B_1 of the KEM to violate the assumed security against adaptive chosen ciphertext.

Adversary B_1 is constructed by running adversary A. We respond to A's queries as follows.

- When A calls any oracle, bar its decryption or challenge encryption oracles, then B_1 simply relays these queries to its own equivalent oracle.
- To respond to A's decryption oracle query for an identity $\mathtt{ID}$ and a ciphertext $(\mathtt{c}_1, \mathtt{c}_2)$ before A has queried its challenge encryption oracle, B_1 proceeds as follows. It first obtains $\mathtt{k}$ by calling its own decapsulation oracle with $\mathtt{c}_1$. If $\mathtt{k} = \perp$ then B_1 replies to A with $\perp$. Otherwise it proceeds to use $\mathtt{k}$ to decrypt $\mathtt{c}_2$ and relays the result to A.
- When A calls its challenge encryption oracle with identities $\mathtt{ID}_i^*$ for $i = 1, \ldots, m$ and messages $(\mathtt{m}_0, \mathtt{m}_1)$, algorithm B_1 first calls its own challenge encryption oracle with $(\mathtt{ID}_i^*)_{i=1}^m$ to obtain $(\mathtt{k}^\dagger, (\mathtt{c}_{i,1}^*)_{i=1}^m)$. It then chooses a bit d at random and computes $\mathtt{c}_2^* \leftarrow \mathbb{E}_{\mathtt{DEM}}(\mathtt{m}_d, \mathtt{k}^\dagger)$. Finally, it responds to A with $(\mathtt{c}_{i,1}^*, \mathtt{c}_{i,2}^*)_{i=1}^m$.
- To respond to A's decryption oracle query for an identity $\mathtt{ID}$ and a ciphertext $(\mathtt{c}_1, \mathtt{c}_2)$ after A has queried its challenge encryption oracle, B_1 proceeds as follows.
  - If $(\mathtt{ID}, \mathtt{c}_1) \neq (\mathtt{ID}_i^*, \mathtt{c}_{i,1}^*)$ then it uses the same procedure that it used before A's call to its challenge encryption oracle.
  - Otherwise, B_1 uses $\mathtt{k}^\dagger$ to decrypt $\mathtt{c}_2$ and relays the result to A.

At the end of the simulation, A outputs a bit d'. If $d' = d$, algorithm B_1 outputs 1, otherwise it outputs 0.

Let b be the internal bit of B_1's challenge oracle which B_1 seeks to determine and let b' be the bit output by B_1. By construction we see that when $b = 1$, so $\mathtt{k}^\dagger$ is the key encapsulated within $\mathtt{c}_{i,1}^*$, algorithm A is run exactly as it would be run in $\mathtt{Game}_1$. This means that

$$\Pr[S_1] = \Pr[d' = d | b = 1] = \Pr[b' = 1 | b = 1],$$

where, from the above simulation, d is A's challenge bit and d' is its guess. Also, when $b = 0$, so a random key $\mathtt{k}'$ is used in the generation of the challenge ciphertext, A is run exactly as it would be in $\mathtt{Game}_2$. This means that:

$$\Pr[S_2] = \Pr[d' = d | b = 0] = \Pr[b' = 1 | b = 0].$$

The result follows from the above two equations and the definitions of security for m-SK-IBKEMs when one observes that

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK-IBKEM}}(B_1) = |2\Pr[b' = b] - 1| = |\Pr[b' = 1 | b = 1] - \Pr[b' = 1 | b = 0]|.$$

□

Lemma 4.2. *There is a PPT algorithm B_2, whose running time is essentially the same as that of A, such that*

$$|\Pr[S_2] - 1/2| = \frac{1}{2}\mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2).$$

Proof. To construct such a B_2 we simply run A as it would be run in game $\mathtt{Game}_2$. We run the m-SK-IBKEM's key generation algorithm so we can respond to A's queries before it calls its challenge encryption oracle. When A calls its challenge encryption oracle with identity $\mathtt{ID}^*_i$ for $i = 1, \ldots, m$ and messages $(\mathtt{m}_0, \mathtt{m}_1)$ we simply relay $(\mathtt{m}_0, \mathtt{m}_1)$ to the challenge encryption oracle of B_2 to obtain $\mathtt{c}^*_{i,2}$. We then run the key encapsulation mechanism to obtain $(\mathtt{k}, \mathtt{c}_{i,1})$ we discard $\mathtt{k}$ and set $\mathtt{c}^*_{i,1} := \mathtt{c}_{i,1}$. Finally we return $(\mathtt{c}^*_{i,1}, \mathtt{c}^*_{i,2})_{i=1}^m$ to A. We continue to respond to A's queries as before except if it a makes decapsulation query on $\mathtt{ID}^*_i$ and $(\mathtt{c}^*_{i,1}, \mathtt{c}_2)$ for some $\mathtt{c}_2$. In this instance we query B_2's decryption oracle with $\mathtt{c}_2$ and relay the response to A.

In this simulation A is run by B_2 in exactly the same manner as the former

would be run in game $\texttt{Game}_2$; moreover, $\Pr[S_2]$ corresponds exactly to the probability that B_2 correctly determines the hidden bit of its challenge encryption oracle since B_2 outputs whatever A outputs. The result follows. □

4.3 Two m-SK-IBKEMs

In this section we present two m-SK-IBKEMs, one of which is a construction that simply uses an identity-based encryption. The second scheme is non-trivial in the sense that it provides a significant improvement in efficiency.

4.3.1 A Simple m-SK-IBKEM

One might think that an obvious way to construct an m-SK-IBKEM would be to use m instances of an IBKEM. However, this is not valid since it would not guarantee that the same key $\texttt{k}$ is encapsulated to all users. In fact, this would imply using m parallel instances of the associated DEM, thereby subverting the principle of efficient hybrid encryption to multiple users.

An m-SK-IBKEM can be easily built using a generic construction similar to that used in [83], whereby an IND-CCA secure public-key encryption algorithm is used to build a secure mKEM. In the identity-based setting, one would take a secure IBE scheme, for instance Boneh and Franklin's FullIdent scheme [23], as the starting point. The resulting m-SK-IBKEM construction operate as follows.

$\mathbb{E}_{m-\texttt{SK}-\texttt{IBKEM}}(\texttt{ID}_1,\ldots,\texttt{ID}_m,\texttt{Mpk})$

- $\texttt{m} \leftarrow \mathcal{M}_{\texttt{IBE}}(\texttt{Mpk})$
- $\texttt{k} \leftarrow \text{KDF}(\texttt{m})$
- For $i = 1,\ldots,m$
  $\texttt{c}_i \leftarrow \mathbb{E}_{\texttt{IBE}}(\texttt{m},\texttt{ID}_i,\texttt{Mpk})$
- Return $(\texttt{k},\texttt{c}_1,\ldots,\texttt{c}_m)$

$\mathbb{D}_{m-\texttt{SK}-\texttt{IBKEM}}(\texttt{c}_i,D_i)$

- $\texttt{m} \leftarrow \mathbb{D}_{\texttt{IBE}}(\texttt{c}_i,D_i)$
- If $\texttt{m} = \perp$ return $\perp$
- $\texttt{k} \leftarrow \text{KDF}(\texttt{m})$
- Return $\texttt{k}$

Here KDF is a key derivation function that maps the message space of the identity-based encryption algorithm to the key space of the DEM.

The security proof in [83] for the generic mKEM construction builds on the work by Bellare et al. [13], who established that a public-key encryption algorithm which is secure in the single-user setting is also secure when multiple users are involved. Therefore proving the security of the previous construction will require extending Bellare et al.'s results to the identity-based setting. Since our main result in this chapter is a more efficient scheme, we do not further debate the security of this simple construction.

4.3.2 An Efficient m-SK-IBKEM from Bilinear Maps

The scheme we propose in this section is inspired by Smart's OR constructions in [82]. Our construction relies on bilinear groups as defined in Section 2.2.2 and uses a parameter generator $\mathbb{G}_{m-\mathtt{SK-IBKEM}}(1^\kappa)$ similar to that of most identity-based schemes. On input a security parameter 1^κ, it outputs the master public key $\mathtt{Mpk}$ and the master secret key $\mathtt{Msk}$, where $\mathtt{Msk} \leftarrow \mathbb{Z}_p^*$ and $\mathtt{Mpk} \leftarrow \mathtt{Msk} \cdot P_1$.

Private key extraction is also performed in the usual way. On input an identity $\mathtt{ID}$ and the master secret key $\mathtt{Msk}$, the extraction algorithm $\mathbb{X}_{m-\mathtt{SK-IBKEM}}$ returns $D \leftarrow \mathtt{Msk} \cdot Q_{\mathtt{ID}}$, where $Q_{\mathtt{ID}} = H_1(\mathtt{ID})$ with $H_1 : \{0,1\}^* \rightarrow G_2$ a cryptographic hash function.

We also choose a cryptographic hash function H_2 that maps elements in $G_T \times G_1 \times G_1$ to the key space of the DEM. The key encapsulation and decapsulation algorithms work as follows. Here P_2 is a generator of G_2.

$\mathbb{E}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{ID}_1,\ldots,\mathtt{ID}_m,\mathtt{Mpk})$

- $r \leftarrow \mathbb{Z}_q^*$; $s \leftarrow \mathbb{Z}_q^*$
- $U_r \leftarrow rP_1$; $U_s \leftarrow sP_1$
- For $i = 1,\ldots,m$
  $U_i \leftarrow r(H_1(\mathtt{ID}_i) - sP_2)$
  $\mathtt{c}_i \leftarrow (U_r, U_s, U_i)$
- $T \leftarrow e(\mathtt{Mpk}, rsP_2)$
- $\mathtt{k} \leftarrow H_2(T, U_r, U_s)$
- Return $(\mathtt{k}, \mathtt{c}_1, \ldots, \mathtt{c}_m)$

$\mathbb{D}_{m-\mathtt{SK}-\mathtt{IBKEM}}(\mathtt{c}_i, D_i)$

- Parse $(U_r, U_s, U_i) \leftarrow \mathtt{c}$
- $t \leftarrow e(U_r, D_i)$
- $t' \leftarrow e(\mathtt{Mpk}, -U_i)$
- $T \leftarrow t \cdot t'$
- $\mathtt{k} \leftarrow H_2(T, U_r, U_s)$
- Return $\mathtt{k}$

Theorem 4.2 describes our result on the security of the above *m*-SK-IBKEM with respect to the security model defined in Section 4.1.4. The technique we use in the proof might allow one to establish the security of a suitable modification of the scheme in [82], for which no proof is provided.

Theorem 4.2. *The m-SK-IBKEM construction described above is IND-CCA secure under the hardness of GBDH problem in the random oracle model. Specifically, for any PPT algorithm A that breaks this m-SK-IBKEM, there exists a PPT algorithm B with*

$$\mathrm{Adv}^{\mathtt{IND}-\mathtt{CCA}}_{m-\mathtt{SK}-\mathtt{IBKEM}}(A) \leq 2q_T^m \cdot \mathrm{Adv}^{\mathtt{GBDH}}_{\Gamma}(B, q_D^2 + 2q_D q_2 + q_2),$$

where q_1, q_2, q_X and q_D denote the maximum number of queries the adversary places to the H_1, H_2, private key extraction and decapsulation oracles respectively, and $q_T = q_1 + q_X + q_D + m$ is the maximum number of queries placed to H_1.

Proof. Let S be the event that A wins the IND-CCA adversarial game. Let also (T^*, U_r^*, U_s^*) denote the value that must be passed to H_2 to obtain the correct key

encapsulated in the challenge ciphertext. Then, we have

$$\Pr[S] = \Pr[S \wedge \texttt{Ask}] + \Pr[S \wedge \neg\texttt{Ask}] = \Pr[S \wedge \texttt{Ask}] + \frac{1}{2}, \tag{4.1}$$

where Ask denotes the event that A queries H_2 with (T^*, U_r^*, U_s^*) during the simulation. This follows from the fact that if A does not place this query, then it can have no advantage.

We argue that in the event $S \wedge \texttt{Ask}$, there is an adversary B that uses A to solve the GBDH problem with probability q_T^{-m} while making at most $q_D^2 + 2q_D q_2 + q_2$ queries to the DBDH oracle. We implement algorithm B as follows.

For a given GBDH problem instance $(\Gamma, aP_2, bP_2, cP_2)$ the strategy is to embed the problem into the *m*-SK-IBKEM challenge presented to the adversary. Namely we construct the simulation so that the value of the pairing associated with the challenge encapsulation is $T^* = e(P_1, P_2)^{abc}$. Hence, when A places a query with the correct value of T^* to H_2, it provides us with the solution to the GBDH problem instance.

We pass Γ and $\texttt{Mpk} = cP_1 = \rho(cP_2)$, where ρ is the isomorphism defined in Section 2.2.2, on to A as the domain parameters. We also maintain three lists that allow us to provide consistent answers to A's oracle calls: $L_1 \subset \{0,1\}^* \times G_2 \times \mathbb{Z}_p$; $L_2 \subset G_T \times G_1 \times G_1 \times \mathcal{K}_{m-\texttt{SK}-\texttt{IBKEM}}(\texttt{Mpk})$; and $L_D \subset \{0,1\}^* \times G_1 \times G_1 \times \mathcal{K}_{m-\texttt{SK}-\texttt{IBKEM}}(\texttt{Mpk})$.

Initially, we generate a list of m random indexes $i_1^*, \ldots, i_m^*$, with $1 \leq i_k^* \leq q_T$. These indexes will determine the set of m identities which B is guessing the adversary will include in its challenge identities. These indexes determine the way in which the hash function H_1 will be simulated.

H_1 queries on $(\texttt{ID}_i)$: On the i-th query to H_1, we check whether i matches one of the indexes i_k^*. If this is the case, we generate random $r_{i_k^*} \in \mathbb{Z}_q^*$, return $Q_{\texttt{ID}_{i_k^*}} =$

$r_{i_k^*}P_2 + bP_2$ and add $(\mathtt{ID}_{i_k^*}, Q_{\mathtt{ID}_{i_k^*}}, r_{i_k^*})$ to L_1. Otherwise, we generate random $r_i \in \mathbb{Z}_q^*$, return $Q_{\mathtt{ID}_i} = r_i P_2$ and add $(\mathtt{ID}_i, Q_{\mathtt{ID}_i}, r_i)$ to L_1.

Algorithm B requires that at the end of phase one, the adversary outputs a list of identities $I\mathcal{D}^* = (\mathtt{ID}_1^*, \ldots, \mathtt{ID}_m^*)$ such that $\mathtt{ID}_k^* = \mathtt{ID}_{i_k^*}$. This happens with probability at least q_T^{-m}. If this is not the case, the simulation fails. Otherwise, the *m*-SK-IBKEM challenge encapsulation is created as

$$\begin{aligned} U_r &\leftarrow \rho(aP_2) \\ U_s &\leftarrow \rho(bP_2) \\ U_k &\leftarrow r_{i_k^*} aP_2 \\ \mathtt{c}_k^* &\leftarrow (U_r, U_s, U_k) \end{aligned}$$

for each challenge $\mathtt{ID}_k^*$. Notice that this is a well-formed encapsulation:

$$U_k = a(Q_{\mathtt{ID}_k^*} - bP_2) = a(r_{i_k^*}P_2 + bP_2 - bP_2) = r_{i_k^*} aP_2.$$

When A outputs $I\mathcal{D}^*$, algorithm B responds with $\mathbf{c}^* = (\mathtt{c}_1^*, \ldots, \mathtt{c}_m^*)$ constructed as above plus a random challenge key $\mathtt{k}^*$.

Given that A is not allowed to obtain the private keys associated with the identities in the challenge, the extraction oracle is easily simulated as follows.

Extraction queries on $(\mathtt{ID}_j)$**:** On input $\mathtt{ID}_j$, we obtain $Q_{\mathtt{ID}_j}$ by calling H_1 first. If $Q_{\mathtt{ID}_j}$ is of the form $r_i P_2$, we look in L_1 for the corresponding r_i and return $D_{\mathtt{ID}_i} = r_i(cP_2)$. If during the first stage of the game, A should query for the extraction of a private key corresponding to a $Q_{\mathtt{ID}}$ of the form $r_{i_k^*}P_2 + bP_2$, the simulation would fail.

Finally, the decapsulation and H_2 oracles are simulated as follows.

Decapsulation queries on $(\mathtt{ID}_\ell, U_{r_\ell}, U_{s_\ell}, U_\ell)$**:** On queries in which we can calculate a value T_ℓ, we do so, and call H_2 with parameters $(T_\ell, U_{r_\ell}, U_{s_\ell})$ to obtain the return value. This will happen when $\mathtt{ID}_\ell$ does not appear in $I\mathcal{D}^*$, in which case we query the private key extraction oracle, and compute T_ℓ as in a standard decapsulation:

$$T_\ell = e(U_{r_\ell}, D_{\mathtt{ID}_\ell}) e(\rho(cP_2), -U_\ell).$$

When $\mathtt{ID}_\ell$ is one of the identities in $I\mathcal{D}^*$ we first search L_2 for an entry such that $U_{r_\ell} = U_{r_n}$, $U_{s_\ell} = U_{s_n}$ and the DBDH oracle returns 1 when queried with the tuple $(U_{r_\ell}, U_{s_\ell}, cP_2, T_n)$. If this entry exists, we return the corresponding $\mathtt{k}_n$. If not, we search L_D for a tuple matching the parameters $(\mathtt{ID}_\ell, U_{r_\ell}, U_{s_\ell})$. If it exists, we return the associated key $\mathtt{k}_\ell$. Otherwise, we generate a random $\mathtt{k}_\ell$, return it and update L_D by adding the entry $(\mathtt{ID}_\ell, U_{r_\ell}, U_{s_\ell}, \mathtt{k}_\ell)$.

H_2 **queries on** $(T_\ell, U_{r_\ell}, U_{s_\ell})$**:** We first check if T_ℓ is the solution we are looking for by calling the DBDH oracle with $(aP_1, bP_1, cP_2, T_\ell)$. If it returns 1, the solution is found, we output T_ℓ and terminate. Otherwise, we proceed to search L_2 for a tuple matching the parameters $(T_\ell, U_{r_\ell}, U_{s_\ell})$. If it exists, we return the corresponding key $\mathtt{k}_\ell$. If not, we search L_D for an entry such that $U_{r_n} = U_{r_\ell}$, $U_{s_n} = U_{s_\ell}$ and the DBDH oracle returns 1 when queried with $(U_{r_n}, U_{s_n}, cP_2, T_n)$. If this entry exists, we return the corresponding $\mathtt{k}_\ell$. Otherwise, we generate a random $\mathtt{k}_\ell$, return it and update L_2 by appending the entry $(T_\ell, U_{r_\ell}, U_{s_\ell}, \mathtt{k}_\ell)$.

To complete the proof, it suffices to notice that B will not fail if the indexes of the identities in $I\mathcal{D}^*$ chosen by A at the end of the first round match the random indexes chosen at the beginning. Since this happens with probability at least q_T^{-m},

we conclude

$$\Pr[S \wedge \mathtt{Ask}] \leq q_T^m \cdot \mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B, q_D^2 + 2q_D q_2 + q_2). \tag{4.2}$$

The total number of DBDH oracle calls, $q_D^2 + 2q_D q_2 + q_2$, can be calculated from the way we simulate the decapsulation and H_2 oracles. The theorem now follows from (4.1) and (4.2). □

We note that the tightness of the security reduction in Theorem 4.2 decreases exponentially with the number of recipients. For practical purposes, and in line with the discussions on practice-oriented provable security in [12], it is important to emphasise that our result provides theoretical security guarantees only in scenarios where messages are encrypted to a small number of recipients. A slightly better result, although still exponential in m, can be obtained by optimising the simulation in the proof above. We note, however, that the above scheme can be proved secure in the selective-ID model with a tighter security reduction. We leave it as an open problem to find a better security reduction for this type of scheme.

4.3.3 Efficiency Considerations

Execution time and bandwidth usage are two important factors affecting the efficiency of a KEM. In this section we present a comparison, with respect to these factors, between the simple scheme described in Section 4.3.1 and the scheme proposed in Section 4.3.2.

The former scheme requires m identity-based encryption computations for encapsulation, and one identity-based decryption computation for decapsulation. Assuming that we use the FullIdent-1 scheme of Boneh and Franklin as described in Section 2.3.5, m pairing computations for encapsulation and another for decapsulation will be needed. On the other hand, as it can be seen from the description of the

latter scheme, it takes only one pairing computation to encapsulate and two more to decapsulate. Note that one can essentially eliminate the pairing computation needed in encapsulation by pre-computing $e(\texttt{Mpk}, P_2)$.

The bandwidth usage of the scheme in Section 4.3.2 is $2\ell_1 + \ell_2$ bits for each recipient, where ℓ_1 and ℓ_2 are the bit-lengths of the representations of elements of G_1 and G_2. Using the FullIdent-1 scheme, the simpler scheme uses $\ell_1 + 2\ell_K$ bits per user where ℓ_K is the length of the session key. For example, using super-singular elliptic curves over characteristic three we could take $\ell_k = 128$ and $\ell_1 = \ell_2 \approx 190$ bits. This means a bandwidth usage of 570 bits for the second scheme versus 446 bits for the first one.

To conclude this discussion we note the m-fold gain in computational speed obtained in Section 4.3.2 is achieved at the cost of a small increase in bandwidth usage, and also by reducing the security of the scheme to the hardness of the GBDH problem, which is a stronger assumption than the hardness of the CBDH problem used in [23]. Furthermore the security reduction of this scheme (in non-selective-ID model) is exponentially slack.

Chapter 5

Randomness-Reuse

In this chapter we investigate the general properties of public-key schemes which allow for computational or bandwidth saving in their associated multi-recipient schemes. We extend the generic framework of reproducibility for reuse of randomness in multi-recipient encryption schemes as proposed by Bellare et al. [14]. A new notion of *weak reproducibility* captures not only encryption schemes which are (fully) reproducible under the criteria given in the previous work, but also a class of efficient schemes which can only be used in the single-message setting. In particular, we are able to capture the single-message schemes suggested by Kurosawa [60], which are more efficient than the direct adaptation of the multiple-message schemes studied by Bellare et al. Our study of randomness-reuse in key encapsulation mechanisms provides an additional argument for the relevance of these results: by taking advantage of our weak reproducibility notion, we are able to generalise and improve multi-recipient KEM constructions found in the literature. We also propose an efficient multi-recipient KEM provably secure in the standard model. We conclude the work on public-key schemes by proposing a notion of *direct reproducibility* which enables tighter security reductions. We then take a brief look at the extensions of our work to the identity-based settings and

analyse some well-known schemes. We conclude this chapter by proposing some open problems for further research. The main part of the work in this chapter first appeared in [10] and represents joint work with Manuel Barbosa.

5.1 Weak Reproducibility

Recall from Section 4.1.1 that we may build the associated *m*-MR-PKE scheme of a PKE scheme by running multiple parallel instances of the base encryption algorithm. If the randomness tapes in all instantiations are identical, the resulting *m*-MR-PKE scheme is called *randomness reusing*.

It is shown in [14] that an IND-atk secure PKE scheme satisfying the definition below can be used to construct an IND-atk secure *m*-MR-PKE scheme (see Section 4.1.1 for the security definition), even reusing randomness across multiple instances. This result is interesting in itself, as it constitutes a generalisation of a proof strategy which can be repeated, almost without change, for all schemes satisfying the reproducibility test. This is a hybrid argument where an *m*-MR-PKE attacker is used to construct an attacker against the base scheme. The reproducibility algorithm generalises the functionality required to extend a challenge in the single-user PKE security game, to construct a complete challenge for the *m*-MR-PKE security game.

Definition 5.1. *A PKE scheme is* fully reproducible *if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1.

1. $(\mathtt{SK},\mathtt{PK}),(\mathtt{SK}',\mathtt{PK}') \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$
2. $r \leftarrow \mathcal{R}_{\mathtt{PKE}}(I)$; $\mathtt{m},\mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{PKE}}(I)$
3. $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m},\mathtt{PK};r)$
4. *If* $R(\mathtt{PK},\mathtt{c},\mathtt{m}',\mathtt{PK}',\mathtt{SK}') = \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}',\mathtt{PK}';r)$ *return* 1 *else return* 0

As seen in Section 4.1.1, the single-message security model for an *m*-SM-PKE is simpler than the full multi-message model. Therefore it is conceivable that a wider range of PKE schemes can be used to construct secure *m*-SM-PKEs, namely efficient randomness reusing ones. Hence, we are interested in defining a less restrictive version of reproducibility that permits determining whether a PKE scheme can be safely used in the single-message scenario, even if it does not satisfy the full reproducibility test above. The following definition achieves this.

Definition 5.2. *A PKE scheme is* weakly reproducible *(wREP) if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1.

1. $(\mathtt{SK}_1, \mathtt{PK}_1), (\mathtt{SK}_2, \mathtt{PK}_2), (\mathtt{SK}_3, \mathtt{PK}_3) \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$
2. $r \leftarrow \mathcal{R}_{\mathtt{PKE}}(I)$; $\mathtt{m}, \mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{PKE}}(I)$
3. $\mathtt{c}_1 \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}_1; r)$; $\mathtt{c}_2 \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}_2; r)$
4. *If* $R(\mathtt{PK}_1, \mathtt{c}_1, \mathtt{m}, \mathtt{PK}_2, \mathtt{SK}_2) \neq \mathtt{c}_2$ *return* 0
5. *If* $R(\mathtt{PK}_1, \mathtt{c}_1, \mathtt{m}', \mathtt{PK}_3, \mathtt{SK}_3) \neq R(\mathtt{PK}_2, \mathtt{c}_2, \mathtt{m}', \mathtt{PK}_3, \mathtt{SK}_3)$ *return* 0 *else return* 1

Similarly to the original full reproducibility definition, the wREP definition follows from the generalisation of the hybrid argument which allows reducing the security of a randomness reusing *m*-SM-PKE to that of its base scheme. The intuition behind the definition is as follows. We are dealing with single-message schemes. Therefore we only require correct reproduction when the two messages are the same. When the messages are different, we relax the definition and require only that *R* is source-PK independent (condition 5). This property is easy to check.

To see why more schemes might satisfy this definition, note that *R* is not even required to produce a valid ciphertext when the messages are different. In Section 5.2 we analyse specific PKE schemes and give a formal separation argument which establishes that the wREP definition is not redundant: there are schemes

which satisfy this definition and which are not fully reproducible. Conversely, it is easy to check that the following lemma holds, and hence wREP fits in the original reproducibility generalisation.

Lemma 5.1. *If a scheme is fully reproducible then it is also weakly reproducible.*

The following theorem shows that the weak reproducibility property is sufficient to guarantee *m*-SM-PKE security. Our proof uses techniques which are somewhat different from that in [14] and may be of independent interest in other contexts.

Theorem 5.1. *The associated randomness reusing m-SM-PKE scheme of an IND-atk public-key encryption scheme is IND-atk secure if the base PKE scheme is weakly reproducible. More precisely for any PPT adversary A against the m-SM-PKE scheme there are two distinct PPT adversaries B and D against the base PKE scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-atk}}_{m-\mathtt{SM-PKE}}(A) \leq m \cdot \mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{PKE}}(B) + (m-1) \cdot \mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{PKE}}(D).$$

Proof. We present the argument for the IND-CPA case. The IND-CCA version is a straightforward extension where simulators use their knowledge of secret keys and external oracles to answer decryption queries.

Consider the following experiment parameterised with an IND-CPA attacker A against the randomness reusing m-SM-PKE scheme, and indexed by a coin b and an integer l such that $0 \leq l \leq m$.

$\mathrm{Exp}_{l,b}(A)$

1. $(\hat{\mathtt{SK}}, \hat{\mathtt{PK}}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$; $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$, $1 \leq i \leq m$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$
3. $\hat{\mathtt{c}} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_b, \hat{\mathtt{PK}})$
4. $\mathtt{c}_i \leftarrow R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_i, \mathtt{SK}_i)$, for $1 \leq i \leq l$
5. $\mathtt{c}_i \leftarrow R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_i, \mathtt{SK}_i)$, for $l+1 \leq i \leq m$
6. $b' \leftarrow A_2(\mathtt{c}_1, \ldots, \mathtt{c}_m, \mathrm{st})$
7. Return b'

Recall from the wREP definition that R performs perfect reproduction when the input message is the same as that inside the input ciphertext. Hence:

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{m-\mathtt{SM}-\mathtt{PKE}}(A) = |\Pr[\mathrm{Exp}_{m,1}(A) = 1] - \Pr[\mathrm{Exp}_{0,0}(A) = 1]|.$$

This follows from the advantage definition, and the fact that when $(l,b) = (m,1)$, ciphertext $\hat{\mathtt{c}}$ will encapsulate $\mathtt{m}_1$, and all challenge ciphertexts are reproduced with $\mathtt{m}_1$, which gives rise to a valid multi-recipient ciphertext encapsulating $\mathtt{m}_1$. The same happens for $\mathtt{m}_0$, when $(l,b) = (0,0)$.

We now define an algorithm B which tries to break the base PKE using A.

$B_1(\overline{\mathtt{PK}})$

1. Select l at random such that $1 \leq l \leq m$
2. $(\mathtt{SK}_l, \mathtt{PK}_l) \leftarrow (\perp, \overline{\mathtt{PK}})$
3. $(\mathtt{SK}_i, \mathtt{PK}_i) \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$, for $1 \leq i \leq m$ and $i \neq l$
4. $(\mathtt{m}_0, \mathtt{m}_1, \mathrm{st}) \leftarrow A_1(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$
5. Return $(\mathtt{m}_0, \mathtt{m}_1, (\mathtt{m}_0, \mathtt{m}_1, l, \overline{\mathtt{PK}}, (\mathtt{PK}_1, \mathtt{SK}_1), \ldots, (\mathtt{PK}_m, \mathtt{SK}_m), \mathrm{st}))$

$B_2(\overline{\mathtt{c}}, (\mathtt{m}_0, \mathtt{m}_1, l, \overline{\mathtt{PK}}, (\mathtt{PK}_1, \mathtt{SK}_1), \ldots, (\mathtt{PK}_m, \mathtt{SK}_m), \mathrm{st}))$

1. $\mathtt{c}_l \leftarrow \overline{\mathtt{c}}$
2. $\mathtt{c}_i \leftarrow R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_i, \mathtt{SK}_i)$, for $1 \leq i \leq l-1$
3. $\mathtt{c}_i \leftarrow R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_i, \mathtt{SK}_i)$, for $l+1 \leq i \leq m$
4. $\hat{b} \leftarrow A_2(\mathtt{c}_1, \ldots, \mathtt{c}_m, \mathrm{st})$
5. Return $\hat{b}$

To continue the proof, we will require the following two lemmas, which we shall prove shortly.

Lemma 5.2. *For $1 \leq l \leq m-1$, and for any PPT adversary A, there is an adversary D such that*

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(D) = |\Pr[\mathrm{Exp}_{l,1}(A) = 1] - \Pr[\mathrm{Exp}_{l,0}(A) = 1]|.$$

Lemma 5.3. *For $1 \leq i \leq m$, the output of algorithm B and that of $\mathtt{Exp}_{l,b}(A)$ are related as follows.*

$$\begin{aligned} \Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 1] &= \Pr[\mathrm{Exp}_{i,1}(A) = 1] \\ \Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 0] &= \Pr[\mathrm{Exp}_{i-1,0}(A) = 1]. \end{aligned}$$

Here $\overline{b}$ is the hidden bit in $\overline{\mathtt{c}}$.

Let us now analyse overall the probability that B returns 1, conditional on the value of the hidden challenge bit $\overline{b}$. Since B chooses l uniformly at random:

$$\begin{aligned} \Pr[\hat{b} = 1 | \overline{b} = 1] &= \frac{1}{m} \sum_{i=1}^{m} \Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 1] \\ \Pr[\hat{b} = 1 | \overline{b} = 0] &= \frac{1}{m} \sum_{i=1}^{m} \Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 0]. \end{aligned}$$

Taking advantage of Lemma 5.3, we can rewrite these as:

$$\begin{aligned}\Pr[\hat{b}=1|\bar{b}=1] &= \frac{1}{m}\sum_{i=1}^{m}\Pr[\mathtt{Exp}_{i,1}(A)=1]\\ \Pr[\hat{b}=1|\bar{b}=0] &= \frac{1}{m}\sum_{i=1}^{m}\Pr[\mathtt{Exp}_{i-1,0}(A)=1].\end{aligned}$$

Subtracting the previous equations and rearranging the terms, we get

$$\begin{aligned}m(\Pr[\hat{b}=1|\bar{b}=1]-\Pr[\hat{b}=1|\bar{b}=0]) &-\\ (\sum_{i=1}^{m-1}\Pr[\mathtt{Exp}_{i,1}(A)=1] &- \sum_{i=1}^{m-1}\Pr[\mathtt{Exp}_{i,0}(A)=1])\\ &= \Pr[\mathtt{Exp}_{m,1}(A)=1]-\Pr[\mathtt{Exp}_{0,0}(A)=1].\end{aligned}$$

Considering the absolute values of both sides and using Lemma 5.2, we can write

$$m\cdot \mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(B)+(m-1)\cdot \mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(D)\geq \mathrm{Adv}^{\mathtt{IND-CPA}}_{m-\mathtt{SM-PKE}}(A).$$

□

We now prove the required lemmas.

Proof. (Lemma 5.2) We build an algorithm $D_l=(D_{1,l},D_{2,l})$ which runs A in exactly the same conditions as it is run in $\mathtt{Exp}_{l,b}$, and which can be used to win the IND-CPA game against the base PKE with an advantage which is the same as A's capability of distinguishing between $\mathtt{Exp}_{l,0}$ and $\mathtt{Exp}_{l,1}$.

$D_{1,l}(\overline{\mathtt{PK}})$

1. $(\mathtt{SK}_i,\mathtt{PK}_i)\leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$, for $1\leq i\leq m$
2. $(\mathtt{m}_0,\mathtt{m}_1,\mathtt{st})\leftarrow A_1(\mathtt{PK}_1,\ldots,\mathtt{PK}_m)$
3. Return $(\mathtt{m}_0,\mathtt{m}_1,(\mathtt{m}_0,\mathtt{m}_1,\overline{\mathtt{PK}},(\mathtt{PK}_1,\mathtt{SK}_1),\ldots,(\mathtt{PK}_m,\mathtt{SK}_m),\mathtt{st}))$

$D_{2,l}(\overline{\mathtt{c}}, (\mathtt{m}_0, \mathtt{m}_1, \overline{\mathtt{PK}}, (\mathtt{PK}_1, \mathtt{SK}_1), \ldots, (\mathtt{PK}_m, \mathtt{SK}_m), \mathrm{st}))$

1. $\mathtt{c}_i \leftarrow R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_i, \mathtt{SK}_i)$, for $1 \leq i \leq l$
2. $\mathtt{c}_i \leftarrow R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_i, \mathtt{SK}_i)$, for $l+1 \leq i \leq m$
3. $\hat{b} \leftarrow A_2(\mathtt{c}_1, \ldots, \mathtt{c}_m, \mathrm{st})$
4. Return $\hat{b}$

Algorithm D simply uses the challenge public key $\overline{\mathtt{PK}}$ in place of $\hat{\mathtt{PK}}$ in the experiment, and uses the PKE challenge $\overline{\mathtt{c}}$ in place of $\hat{\mathtt{c}}$. Note that the only visible difference to the definition of $\mathtt{Exp}$ is that D does not know $\overline{\mathtt{SK}}$, which it does not need[1], and that the IND-CPA hidden bit $\overline{b}$ is used in place of b. We can therefore write, for a given value of l:

$$\begin{aligned} \Pr[\hat{b} = 1 | \overline{b} = 1] &= \Pr[\mathtt{Exp}_{l,1}(A) = 1]| \\ \Pr[\hat{b} = 1 | \overline{b} = 0] &= \Pr[\mathtt{Exp}_{l,0}(A) = 1]|, \end{aligned}$$

and consequently

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(D) = |\Pr[\mathtt{Exp}_{l,1}(A) = 1] - \Pr[\mathtt{Exp}_{l,0}(A) = 1]|.$$

□

Proof. (Lemma 5.3) The first case of the lemma states that

$$\Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 1] = \Pr[\mathtt{Exp}_{i,1}(A) = 1].$$

We must show that the probability distribution of the inputs presented to A is exactly the same in the scenarios corresponding to both sides of the equation above. This is trivially true for the public keys that A_1 receives, since all of them are

[1] In the CCA setting the decryption queries are answered using the equivalent outside oracle.

independently generated using the correct algorithm. Regarding the challenge ciphertext that A_2 gets, we start by expanding the values of $(\mathtt{c}_1, \ldots, \mathtt{c}_m)$.

In $\mathtt{Exp}_{i,1}(A)$, we have $\hat{\mathtt{c}} = \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_1, \hat{\mathtt{PK}}; r)$ and

$$\begin{aligned} \mathtt{c}_j &= R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } 1 \leq j \leq i \\ \mathtt{c}_j &= R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } i+1 \leq j \leq m. \end{aligned}$$

Now, in $B_2(\overline{\mathtt{c}}, \overline{\mathtt{st}})$, given that $l = i$ and $\overline{b} = 1$ we have $\overline{\mathtt{c}} = \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_1, \overline{\mathtt{PK}}; r)$ and

$$\begin{aligned} \mathtt{c}_i &= \overline{\mathtt{c}} \\ \mathtt{c}_j &= R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } 1 \leq j \leq i-1 \\ \mathtt{c}_j &= R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } i+1 \leq j \leq m. \end{aligned}$$

To show that the distributions are identical, we split the argument in three parts and fix the values of all random variables, considering the case where the public keys provided to A in both cases are the same, and that the implicit randomness in both $\hat{\mathtt{c}}$ and $\overline{\mathtt{c}}$ is the same r. We show that the resulting challenge ciphertexts in both cases are exactly the same:

- $j = i$: Note that in the second scenario we have $\mathtt{c}_i = \overline{\mathtt{c}}$, while in the first scenario we have $\mathtt{c}_i = R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_i, \mathtt{SK}_i)$. Since $\hat{\mathtt{c}}$ encrypts $\mathtt{m}_1$, the result of R is perfect and equal to $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_1, \mathtt{PK}_i; r) = \overline{\mathtt{c}}$.

- $j < i$: In this range, challenge components are identical in both scenarios: they are perfect reproductions $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_1, \mathtt{PK}_j; r)$, since $\mathtt{m}_1$ is passed to R both in encrypted and plaintext form.

- $j > i$: In this range, challenge components are outputs of R, but in this case we cannot claim that they are identical without resorting to the properties of the wREP algorithm. For different message reproduction, condition 5 of

Definition 5.2 ensures that

$$R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j) = R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j),$$

as required.

This means that the first case of the lemma follows.

We now prove the lemma for the case

$$\Pr[\hat{b} = 1 | l = i \wedge \overline{b} = 0] = \Pr[\mathtt{Exp}_{i-1,0}(A) = 1].$$

The argument is similar to the previous case. We must show that the probability distribution of the inputs presented to A is exactly the same in the scenarios corresponding to both sides of the equation above. This is trivially true for the public keys that A_1 receives, since all of them are independently generated using the correct algorithm. Regarding the challenge ciphertext that A_2 gets, we start by expanding the values of $(\mathtt{c}_1, \ldots, \mathtt{c}_m)$.

In $\mathtt{Exp}_{i-1,0}(A)$, we have $\hat{\mathtt{c}} = \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_0, \hat{\mathtt{PK}}; r)$ and

$$\begin{aligned} \mathtt{c}_j &= R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } 1 \leq j \leq i-1 \\ \mathtt{c}_j &= R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } i \leq j \leq m. \end{aligned}$$

Now, in $B_2(\overline{\mathtt{c}}, \hat{\mathtt{st}})$, given that $l = i$ and $\overline{b} = 0$ we have $\overline{\mathtt{c}} = \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_0, \mathtt{PK}; r)$ and

$$\begin{aligned} \mathtt{c}_i &= \overline{\mathtt{c}} \\ \mathtt{c}_j &= R(\mathtt{PK}, \overline{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } 1 \leq j \leq i-1 \\ \mathtt{c}_j &= R(\mathtt{PK}, \overline{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_j, \mathtt{SK}_j) \text{ for } i+1 \leq j \leq m. \end{aligned}$$

To show that the distributions are identical, we split the argument in three parts and fix the values of all random variables, considering the case where the public keys

provided to A in both cases are the same, and that the implicit randomness in both $\hat{\mathtt{c}}$ and $\overline{\mathtt{c}}$ is the same r. We show that the resulting challenge ciphertexts in both cases are exactly the same:

- $j = i$: Note that in the second scenario we have $\mathtt{c}_i = \overline{\mathtt{c}}$, while in the first scenario we have $\mathtt{c}_i = R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_0, \mathtt{PK}_i, \mathtt{SK}_i)$. Since $\hat{\mathtt{c}}$ encrypts $\mathtt{m}_0$, the result of R is perfect and equal to $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_0, \mathtt{PK}_i; r) = \overline{\mathtt{c}}$.

- $j < i$: In this range, challenge components are outputs of R, but in this case we cannot claim that they are identical without resorting to the properties of R described in Definition 5.2 for different message reproduction, which ensure that

$$R(\hat{\mathtt{PK}}, \hat{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j) = R(\overline{\mathtt{PK}}, \overline{\mathtt{c}}, \mathtt{m}_1, \mathtt{PK}_j, \mathtt{SK}_j),$$

  as required.

- $j > i$: In this range, challenge components are identical in both scenarios: they are perfect reproductions $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}_0, \mathtt{PK}_j; r)$, since $\mathtt{m}_0$ is passed to R both in encrypted and plaintext form.

This means that the second case of the lemma follows. □

5.2 Kurosawa's Efficient Schemes

In this section we analyse modified versions of ElGamal and Cramer–Shoup encryption schemes briefly mentioned by Kurosawa [60] as a way to build efficient single-message multiple-recipient public-key encryption schemes. These schemes permit establishing a separation between the original reproducibility notion proposed by Bellare et al. in [14] and the one we introduced in the previous section.

5.2.1 Modified ElGamal Encryption Scheme

The *modified ElGamal* encryption scheme is similar to the ElGamal encryption scheme and operates as follows. The key generation algorithm $\mathbb{G}_{\mathtt{PKE}}(I)$ on input $I := (p,P)$ returns the key-pair $(\mathtt{SK},\mathtt{PK}) = (1/x, xP)$ for $x \leftarrow \mathbb{Z}_p^*$. The encryption algorithm $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m},\mathtt{PK};r)$ returns the ciphertext $(U,V) := (r(xP), \mathtt{m}+rP)$ for $r \leftarrow \mathbb{Z}_p^*$. The decryption algorithm $\mathbb{D}_{\mathtt{PKE}}(U,V,1/x)$ returns the message $\mathtt{m} := V - (1/x)U$.

Theorem 5.2 establishes the security of the modified ElGamal scheme as well as its weak reproducibility property. Theorem 5.3 shows that modified ElGamal establishes a separation between the notions of full and weak reproducibility.

Theorem 5.2. *Modified ElGamal is (1) IND-CPA secure under the decisional Diffie–Hellman assumption. In other words for any adversary A against the scheme there is an algorithm B which solves the DDH problem such that* $\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(A) \leq 2\mathrm{Adv}^{\mathtt{DDH}}_{\Gamma}(B)$*; and (2) weakly reproducible.*

Proof. (1) The proof is similar to that for the ElGamal encryption scheme. We use a sequence of two games $\mathtt{Game}_0$ and $\mathtt{Game}_1$. We denote by S_i the event that an adversary A succeeds in guessing the secret bit in $\mathtt{Game}_i$. Let $\mathtt{Game}_0$ be the original IND-CPA security game. We have

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(A) = |2\Pr[S_0] - 1|.$$

In game $\mathtt{Game}_1$ we make a single change by constructing the challenge setting $U = sP$ for a random s. An adversary can have no advantage in $\mathtt{Game}_1$, as the message is encrypted with a one-time padded, i.e. $\Pr[S_1] = 1/2$.

We now claim that, for any adversary A of the modified ElGamal scheme, we can construct an algorithm B which solves the DDH problem such that:

$$|\Pr[S_0] - \Pr[S_1]| = \mathrm{Adv}^{\mathtt{DDH}}_{\Gamma}(B).$$

This follows from the observation that in $\texttt{Game}_0$, the tuple $(P, xP, V - \mathtt{m}, U)$ is of the form (P, xP, rP, rxP) while in $\texttt{Game}_1$ it is of the form (P, xP, rP, sP). More precisely, algorithm B on input (P, aP, bP, Q) works as follows. It sets the public key to be aP and runs the adversary on this to get $\mathtt{m}_0$ and $\mathtt{m}_1$. It then sets $(U, V) := (Q, \mathtt{m}_\beta + bP)$ for random bit β, and returns 1 if $\beta = \beta'$ and 0 otherwise. Here β' is the bit returned by A. Now if $Q = abP$, A is being run according to the rules of $\texttt{Game}_0$; and, conversely, if Q is random, the environment is that of $\texttt{Game}_1$. Hence:

$$\mathrm{Adv}^{\mathtt{DDH}}_{\Gamma}(B) = |\Pr[\beta' = \beta | Q = abP] - \Pr[\beta' = \beta | Q = sP]| = |\Pr[S_0] - \Pr[S_1]|.$$

Putting these results together we get:

$$\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(A) = |2\Pr[S_0] - 1| = 2|\Pr[S_0] - \Pr[S_1]| = 2\mathrm{Adv}^{\mathtt{DDH}}_{\Gamma}(B).$$

(2) The weak reproducibility algorithm R on input $(xP, U, V, \mathtt{m}', x'P, 1/x')$ returns $(x'(V - \mathtt{m}'), V)$. We now check that R satisfies the two properties required by the wREP definition. If $\mathtt{m}' = \mathtt{m}$, then $V - \mathtt{m}' = (\mathtt{m} + rP) - \mathtt{m} = rP$ and the output is a valid encryption of $\mathtt{m}'$ under $x'P$ using random coins r. Note also that R's output does not depend on the public key xP and hence the second property is also satisfied. □

Theorem 5.3. *The modified ElGamal encryption is not fully reproducible under the CDH assumption.*

Proof. Let $(P, aP, bP) \in G^3$ denote the CDH problem instance. Our goal is to compute abP. The reproduction algorithm on input $(P, xP, rxP, \mathtt{m} + rP, \mathtt{m}', yP, 1/y)$ outputs $(ryP, \mathtt{m}' + rP)$. We pass to R the input $(aP, P, bP, 1, 1, aP, 1)$ which could be written as $(Q, (1/a)Q, (b/a)Q, 1, 1, Q, 1)$ where $Q = aP$. Here, implicitly, we have $x = 1/a$. From $rx = b/a$ we get $r = b$ and $\mathtt{m} = (-b)Q$. Hence this input is valid.

The first component of the output of R will be therefore $(1 \cdot b)Q = abP$. □

Remark. Note that since R succeeds with probability 1, it will run correctly on the input instance passed to it in the above proof, even though its distribution is distinguishable from those that R takes. If required, it is possible to adjust the input so that it has a distribution which is statistically close to those that R takes. Note also that by changing the generator, we have changed the group scheme which R is working on. This new group scheme is a valid output of the setup procedure of modified ElGamal, and hence R will run correctly.

Remark. Let us also note that modified ElGamal encryption scheme is *insecure* when used in the multi-message scenario. This is because the adversary can choose $\mathtt{m}_{2,0} = \mathtt{m}_{2,1} = \mathtt{m}$ for the second user and then recover the mask rP by calculating $\mathtt{c}^*_{2,2} - \mathtt{m}$. The scheme is, however, secure *without* randomness-reuse [13].

5.2.2 Modified Cramer–Shoup Encryption Scheme

Another construction of an efficient m-SM-PKE hinted at by Kurosawa [60] is based on the CS1a encryption scheme of Cramer and Shoup [37], modified in an analogous manner to the ElGamal encryption scheme as presented in the previous section. In this case, the construction is secure against adaptive chosen ciphertext attacks in the standard model. Modified versions of the other schemes presented in [37] also pass the weak reproducibility test without being fully reproducible. The following scheme, however, is the most efficient among these schemes as it shares $\hat{g}$ as a domain parameter.

We use multiplicative notation for the ease of comparison with the original scheme. The domain parameter is $I := (p, g, \hat{g}, H)$, where g and $\hat{g}$ are generators of a group G of prime order p and H denotes a cryptographic hash function. The key generation algorithm outputs (x_1, x_2, y_1, y_2, z), a random element of $(\mathbb{Z}_p)^5$, as

the secret key and the public key is set to be $(g^{x_1}\hat{g}^{x_2}, g^{y_1}\hat{g}^{y_2}, g^z)$.

Encryption and decryption algorithms work as follows.

$\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK})$

- $(e, f, h) \leftarrow \mathtt{PK}$
- $u \leftarrow \mathbb{Z}_p$
- $\hat{a} \leftarrow \hat{g}^u$
- $b \leftarrow h^u$
- $c \leftarrow \mathtt{m} \cdot g^u$
- $v \leftarrow \mathrm{H}(\hat{a}, b, c)$
- $d \leftarrow e^u f^{uv}$
- Return $(\hat{a}, b, c, d)$

$\mathbb{D}_{\mathtt{PKE}}((\hat{a}, b, c, d), \mathtt{SK})$

- $(x_1, x_2, y_1, y_2, z) \leftarrow \mathtt{SK}$
- $v \leftarrow \mathrm{H}(\hat{a}, b, c)$
- $a \leftarrow b^{1/z}$
- If $a^{x_1+y_1 v}\hat{a}^{x_2+y_2 v} \neq d$ return $\perp$
- $\mathtt{m} \leftarrow c/a$
- Return $\mathtt{m}$

Theorem 5.4. *The modified Cramer–Shoup scheme is weakly reproducible.*

Proof. The weak reproduction algorithm is a natural extension of the one presented for modified ElGamal, returning

$$(\hat{a}, (c/\mathtt{m}')^{z'}, c, (c/\mathtt{m}')^{(x_1'+v'y_1')}\hat{a}^{(x_2'+v'y_2')}),$$

where $v' := H(\hat{a}, (c/\mathtt{m}')^{z'}, c)$. □

We leave open the IND-CCA security of the above scheme under the DDH assumption.

A very important distinction in this case, however, is that the reproduction algorithm produces an output *which may not be a valid ciphertext* in the sense that it cannot be an output of the encryption algorithm. In fact, for different-message reproduction, the encryption algorithm would never be able to produce something

like the resulting ciphertext. The returned output is, however, indistinguishable from a valid ciphertext under the decisional Diffie–Hellman assumption. The fact that the outputs of R may not be identically distributed to the outputs of the encryption algorithm, but merely indistinguishable, implies that the proof strategy presented in [14] does *not* apply for this scheme. On the other hand, note that the technique presented in the proof of Theorem 5.1 covers this and other similar schemes.

5.3 Hybrid Encryption

Practical applications of public-key encryption are based on the hybrid paradigm as described in Section 2.4.1. Sharing randomness across multiple instances of a KEM may be justified, as before, as a means to achieve computational savings when performing batch operations. In this section we study randomness-reuse for building efficient m-MR-KEMs and m-SK-KEMs (see Section 4.1.2 for definitions), a problem which has not been formally addressed in previous work.

The KEM primitive takes the recipient's public key as the single parameter to the encapsulation algorithm. In particular, unlike what happens in PKEs, one does not control the value of the encapsulated key: this is internally generated inside the KEM primitive, and its value depends only on the recipient's public key and on the randomness tape of the encapsulation algorithm. In this work we are interested on the role of randomness in building m-SK-KEMs. This leads us to the following categorisation of KEMs.

Definition 5.3. *A KEM is* public key independent *if the following experiment returns* 1 *with probability* 1.

1. $(\mathtt{SK}, \mathtt{PK}), (\mathtt{SK}', \mathtt{PK}') \leftarrow \mathbb{G}_{\mathtt{KEM}}(I)$
2. $r \leftarrow \mathcal{R}_{\mathtt{KEM}}(I)$

3. $(\mathtt{k}, \mathtt{c}) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK}; r)$; $(\mathtt{k}', \mathtt{c}') \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK}'; r)$

4. *If* $\mathtt{k} = \mathtt{k}'$ *return* 1 *else return* 0

Adaptation of the results in [14] to *m*-MR-KEMs is straightforward. The same is not true, however, for *m*-SK-KEMs. To justify why this is the case, we present a reproducibility test for KEMs in Definition 5.4. It is a direct adaptation of the reproducibility test for PKEs, considering that there is no message input to the encapsulation algorithm and that this returns also the encapsulated secret key. It can be easily shown that any KEM satisfying this test can be used to construct an efficient *m*-MR-KEM with randomness-reuse.

Definition 5.4. *A KEM is called* reproducible *if there exists a PPT algorithm R such that the following experiment returns* 1 *with probability* 1.

1. $(\mathtt{SK}, \mathtt{PK}), (\mathtt{SK}', \mathtt{PK}') \leftarrow \mathbb{G}_{\mathtt{KEM}}(I)$

2. $r \leftarrow \mathcal{R}_{\mathtt{KEM}}(I)$

3. $(\mathtt{k}, \mathtt{c}) \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK}; r)$; $(\mathtt{k}', \mathtt{c}') \leftarrow \mathbb{E}_{\mathtt{KEM}}(\mathtt{PK}'; r)$

4. *If* $R(\mathtt{PK}, \mathtt{c}, \mathtt{PK}', \mathtt{SK}') = (\mathtt{k}', \mathtt{c}')$ *return* 1 *else return* 0

Unlike *m*-MR-KEMs there does not seem to be a natural way of constructing *m*-SK-KEMs from single-recipient KEMs. The fact that the same key should be encapsulated for all recipients makes public key independent KEMs the only possible candidates to be used as base KEMs. However, any public key independent scheme which satisfies a reproducibility test such as that in Definition 5.4 must be insecure, as anyone would be able to use the reproducibility algorithm to obtain the secret key in an arbitrary ciphertext. In the following section we show how the weak reproducibility notion for PKEs we obtained in Theorem 5.1 actually fills this apparent theoretical gap, as it permits capturing the efficient *m*-SK-KEMs constructions we have found in literature.

5.3.1 Generic Construction of *m*-SK-KEMs

As in Section 4.3.1, one trivial way to build secure randomness reusing *m*-SK-KEMs is to use a secure weakly reproducible encryption scheme, and to set a random message to be the ephemeral key. However, the underlying encryption scheme must have the same security guarantees as those required for the KEM.

A more practical way to build a fully secure *m*-SK-KEM is to use a weaker PKE through the following generic construction, which generalises the mKEM proposed in [83] and extends the construction by Dent in Section 2.4.3. The domain parameters and key generation algorithm are the same as those of the underlying scheme. Encapsulation and decapsulation algorithms are as follows.

$\mathbb{E}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{PK}_1, \ldots, \mathtt{PK}_m)$

- $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{PKE}}(I)$
- $r \leftarrow H(\mathtt{m})$
- For $i = 1 \ldots, m$
  $\mathtt{c}_i \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}_i; r)$
- $\mathbf{c} \leftarrow (\mathtt{c}_1, \ldots, \mathtt{c}_m)$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\mathtt{m})$
- Return $(\mathtt{k}, \mathbf{c})$

$\mathbb{D}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{c}, \mathtt{SK})$

- $\mathtt{m} \leftarrow \mathbb{D}_{\mathtt{PKE}}(\mathtt{c}, \mathtt{SK})$
- If $\mathtt{m} = \perp$ return $\perp$
- $r \leftarrow H(\mathtt{m})$
- If $\mathtt{c} \neq \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r)$ return $\perp$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\mathtt{m})$
- Return $\mathtt{k}$

Here H and KDF are cryptographic hash functions. The security argument for this construction has two parts. The first part establishes the one-way security of the *m*-SK-PKE scheme associated with the base PKE. This follows directly from the weak reproducibility theorem in Section 5.1 and the fact that one-wayness is implied by indistinguishability[2]. The second part builds on the previous result to achieve IND-CCA security in the *m*-SK-KEM setting using Dent's construction

[2]We assume that various message spaces have exponential size in the security parameter.

in Section 2.4.3. In this construction one models the hash function H and KDF as random oracles and shows that the queries placed by any adversary with non-negligible advantage in breaking the m-SK-KEM can be used to invert the one-wayness of the underlying m-SM-PKE scheme.

Theorem 5.5. *The above construction is an IND-CCA secure m-SK-KEM, if the underlying PKE is IND-CPA and weakly reproducible, and if we model H and* KDF *as random oracles. More precisely, for any PPT adversary A against the scheme there is a PPT adversary B against the underlying PKE scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK-KEM}}(A) \leq 2m(q_H + q_K + q_D)\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(B) + \varepsilon,$$

where q_H, q_K and q_D are the number of queries the adversary makes to H, KDF *and decapsulation oracles and* ε *denotes a negligible quantity.*

Proof. Let A denote an IND-CCA adversary against the generic construction with non-negligible advantage. Modelling hash functions as random oracles, we construct an algorithm B with non-negligible advantage in the OW-CPA game for the m-SM-PKE. One-wayness can be easily adapted to multi-recipient schemes. Note that one-wayness of an m-SM-PKE scheme is not necessarily implied by the one-wayness of its base PKE [51]. However, since indistinguishability implies one-wayness and the indistinguishability property is inherited from the base scheme due to the wREP property, we do have that the m-SM-PKE is OW-CPA. The concrete reduction is:

$$\mathrm{Adv}^{\mathtt{OW-CPA}}_{m-\mathtt{SM-PKE}}(B) \leq \mathrm{Adv}^{\mathtt{IND-CPA}}_{m-\mathtt{SM-PKE}}(C) + \varepsilon_1 \leq m\mathrm{Adv}^{\mathtt{IND-CPA}}_{\mathtt{PKE}}(D) + \varepsilon_2.$$

Here ε_1 and ε_2 are negligible quantities, assuming that the message space has a size super-polynomial in the security parameter. We omit the straightforward details of

the proof of the above inequality.

On receiving the m public keys for the OW-CPA game, B passes these values on to algorithm A. During A's first stage, algorithm B replies to A's oracle queries as follows:

- H queries: B maintains a list $L \subseteq \mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I) \times \mathcal{R}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$ which contains at most q_H pairs $(\mathtt{m}, r)$. On input of $\mathtt{m}$, if $(\mathtt{m}, r) \in L$ then B returns r, otherwise it selects r at random from the appropriate randomness space, appends $(\mathtt{m}, r)$ to the list and returns r.

- KDF queries: B maintains a list $L_K \subseteq \mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I) \times \mathcal{K}_{m-\mathtt{SK}-\mathtt{KEM}}(I)$ which contains at most $q_K + q_D$ pairs $(\mathtt{m}, \mathtt{k})$. On input of $\mathtt{m}$, if $(\mathtt{m}, \mathtt{k}) \in L_K$ then B returns $\mathtt{k}$, otherwise it selects $\mathtt{k}$ at random from the appropriate key space, appends $(\mathtt{m}, \mathtt{k})$ to the list and returns $\mathtt{k}$.

- Decapsulation queries: on input $(\mathtt{c}, \mathtt{PK})$, B checks for each $(\mathtt{m}, r) \in L$ if $\mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r) = \mathtt{c}$; if such a pair exists, B calls the KDF simulation procedure above on value $\mathtt{m}$ and returns the result to A. Otherwise B returns $\perp$.

At some point A will complete its first stage and return some state information. At this point, B calls the outside challenge oracle, and obtains a challenge ciphertext $(\mathtt{c}_1, \ldots, \mathtt{c}_m)$ on some unknown $\mathtt{m}^*$. Algorithm B now checks if A has queried for decapsulation on a tuple $(\mathtt{c}_\ell, \mathtt{PK}_\ell)$ during its first stage. If this is the case, algorithm B terminates. Otherwise it generates a random $\mathtt{k}^*$ and provides this to A along with the challenge ciphertext.

In the second stage, B answers A's oracle queries as in stage one. When A terminates, B randomly returns a message from L or L_K.

Now we analyse the probability that this answer is correct.

Algorithm B's execution has no chance of success (event S_B) if it terminates at the end of A's first stage (event T). Therefore:

$$\Pr[S_B] = \Pr[S_B \wedge \neg T] = \Pr[S_B | \neg T] \Pr[\neg T].$$

Note that the challenge encapsulation is independent of A's view in the first stage, so that A could only have queried decapsulation for one of the challenge encapsulations by pure chance. However, the size of the valid encapsulation space for each public key is the same as the message space. This means that the probability that B does not abort is

$$\Pr[\neg T] = 1 - \frac{q_D}{M},$$

where $M = |\mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I)|$.

Given that termination does not take place, B's simulation could be imperfect if one of the following events occurs:

- Event E_1: The adversary places a decapsulation query for a valid ciphertext, and B returns $\perp$.
- Event E_2: The adversary queries H or KDF for the unknown $\mathtt{m}^*$ value.

Event E_1 occurs if A finds a valid ciphertext without querying H to obtain the randomness required to properly construct it. The probability of this is

$$\Pr[E_1] \leq \frac{q_D \gamma_m}{R},$$

where $R = |\mathcal{R}_{m-\mathtt{SM}-\mathtt{PKE}}(I)|$ and $\gamma_m = \gamma_m(I)$ is the least upper bound such that for every m-tuple $(\mathtt{PK}_i)_{i=1}^m$, every $\mathtt{m} \in \mathcal{M}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$, every $j \in \{1, \ldots, m\}$ and every $\mathtt{c} \in \mathcal{C}_{m-\mathtt{SM}-\mathtt{PKE}}(I)$ we have

$$|\{r \in \mathcal{R}_{m-\mathtt{SM}-\mathtt{PKE}}(I) : [\mathbb{E}_{m-\mathtt{SM}-\mathtt{PKE}}(\mathtt{m}, (\mathtt{PK}_i)_{i=1}^m; r)]_j = \mathtt{c}\}| \leq \gamma_m(I).$$

This follows from the fact that, since H is modelled as a random oracle, A can only achieve this by guessing the value of randomness. Moreover, the probability that a given randomness generates a valid ciphertext is at most γ_m/R and there are at most q_D such queries.

Note that we can write

$$\Pr[E_1 \vee E_2] \leq \Pr[E_1] + \Pr[E_2] \leq \frac{q_D \gamma_m}{R} + \Pr[E_2].$$

On the other hand, since A operates in the random oracle model it can have no advantage if event $E_1 \vee E_2$ does not occur. Hence we can write

$$\begin{aligned}
\frac{1}{2}\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK}-\mathtt{KEM}}(A) &= |\Pr[S_A] - 1/2| \\
&= |\Pr[S_A \wedge (E_1 \vee E_2)] + \Pr[S_A \wedge \neg(E_1 \vee E_2)] - 1/2| \\
&\leq \Pr[E_1 \vee E_2] + 1/2 - 1/2 \\
&\leq \Pr[E_2] + \frac{q_D \gamma_m}{R}.
\end{aligned}$$

Now:

$$\begin{aligned}
\mathrm{Adv}^{\mathtt{OW-CPA}}_{m-\mathtt{SM}-\mathtt{PKE}}(B) &= \Pr[S_B] = \Pr[S_B|\neg T](1 - \frac{q_D}{M}) \\
&= \frac{1}{|L| + |L_K|}\Pr[E_2](1 - \frac{q_D}{M}) \\
&\geq \frac{1}{q_H + q_K + q_D}(\Pr[E_2] - \frac{q_D}{M}).
\end{aligned}$$

Rearranging the terms we get:

$$\Pr[E_2] \leq (q_H + q_K + q_D)\mathrm{Adv}^{\mathtt{OW-CPA}}_{m-\mathtt{SM}-\mathtt{PKE}}(B) + \frac{q_D}{M}.$$

Putting the above two results together we obtain the required result:

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{m-\mathtt{SK}-\mathtt{KEM}}(A) \leq 2(q_H + q_K + q_D)\mathrm{Adv}^{\mathtt{OW-CPA}}_{m-\mathtt{SM}-\mathtt{PKE}}(B) + 2q_D(\frac{1}{M} + \frac{\gamma_m}{R}).$$

□

The mKEM in [83] fits the general framework we introduced in this section by instantiating the above construction with the ElGamal encryption scheme. The results in this work permit introducing two interesting enhancements over the mKEM in [83], if the above construction is instantiated with the modified ElGamal scheme:

- Stronger security guarantees disallowing benign malleability. The security model in [83] disallows decapsulation queries on any ciphertext which decapsulates to the same key as that implicit in the challenge.
- More efficient encryption algorithm, saving $m-1$ group operations.

5.3.2 An Efficient m-SK-KEM in the Standard Model

In this section we propose an efficient m-SK-KEM candidate, with conjectured IND-CCA security in the standard model. To the best of our knowledge, it is the first such construction which could achieve this level of security and efficiency. The scheme, which is an adaptation of a KEM proposed by Cramer and Shoup in [37], is as follows.

Once again we use multiplicative notation. The domain parameter is $I := (p, g, \hat{g}, H, \mathrm{KDF})$, where g and $\hat{g}$ are generators of a group G of prime order p, H is a cryptographic hash function and KDF is a key derivation function. The key generation algorithm $\mathbb{G}_{m-\mathtt{SK-KEM}}(I)$ outputs $\mathtt{SK} = (x_1, x_2, y_1, y_2, z)$, a random element of $\mathbb{Z}_p^4 \times \mathbb{Z}_p^*$, as the secret key and $\mathtt{PK} = (e, f, h) = (g^{x_1}\hat{g}^{x_2}, g^{y_1}\hat{g}^{y_2}, g^z)$ as the public key. Encapsulation and decapsulation algorithms described below.

$\mathbb{E}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{PK}_1,\ldots,\mathtt{PK}_m)$

- $u \leftarrow \mathbb{Z}_p$
- $\hat{a} \leftarrow \hat{g}^u$; $b \leftarrow g^u$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\hat{a}, b)$
- For $i = 1,\ldots,m$
  $(e_i, f_i, h_i) \leftarrow \mathtt{PK}_i$
  $a_i \leftarrow h_i^u$
  $v_i \leftarrow H(\hat{a}, a_i)$
  $d_i \leftarrow e_i^u f_i^{uv_i}$
  $\mathtt{c}_i \leftarrow (\hat{a}, a_i, d_i)$
- Return $(\mathtt{k}, \mathtt{c}_1, \ldots, \mathtt{c}_m)$

$\mathbb{D}_{m-\mathtt{SK}-\mathtt{KEM}}(\mathtt{c}, \mathtt{SK})$

- Parse $(\hat{a}, a, d) \leftarrow \mathtt{c}$
- $(x_1, x_2, y_1, y_2, z) \leftarrow \mathtt{SK}$
- $v \leftarrow H(\hat{a}, a)$
- $b \leftarrow a^{1/z}$
- If $b^{x_1+vy_1}\hat{a}^{x_2+vy_2} \neq d$ return $\perp$
- $\mathtt{k} \leftarrow \mathrm{KDF}(\hat{a}, b)$
- Return $\mathtt{k}$

5.4 Tighter Reductions

5.4.1 Direct Reproducibility

In [14] the authors present tighter security reductions for the multi-recipient randomness reusing schemes associated with the ElGamal and Cramer–Shoup encryption schemes. These reductions rely on the random self-reducibility property of the DDH problem [13]. The tighter reductions are achieved by using this property to unfold a single DDH problem instance, so that it can be embedded in the multiple challenge ciphertext components required in the multiple user setting. In these proofs, the extra public keys and challenge ciphertexts required in the reduction are chosen in a controlled manner. For instance, one public key might have a known discrete logarithm with respect to another. The following notion of reproducibility could be viewed as a generalisation of this type of proof strategy for tight reductions.

Definition 5.5. *A PKE scheme is called* directly reproducible *(dREP) if there exists a triple of stateful PPT machines* $R = (R_1, R_2, R_3)$ *such that the following experiment returns* 1 *with probability* 1.

1. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}_{\mathtt{PKE}}(I)$
2. $(\mathtt{PK}', \mathrm{st}) \leftarrow R_1(\mathtt{PK})$
3. $r \leftarrow \mathcal{R}_{\mathtt{PKE}}(I)$; $\mathtt{m} \leftarrow \mathcal{M}_{\mathtt{PKE}}(I)$
4. $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}; r)$; $\mathtt{c}' \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{m}, \mathtt{PK}'; r)$
5. *If* $\mathtt{c}' \neq R_2(\mathtt{c}, \mathrm{st})$ *return* 0
6. *If* $\mathtt{c} \neq R_3(\mathtt{c}', \mathrm{st})$ *return* 0 *else return* 1

We require the distributions of PK *and* PK′ *to be (almost) identical.*

Note that R_1 controls the generation of the public keys and the main reproduction algorithm (R_2) may take advantage of the state information produced by the first algorithm. The existence of the third algorithm is required for the simulation of decryption oracles for CCA-secure schemes and is not needed in the CPA model. It is easy to show that:

Theorem 5.6. *The associated randomness reusing m-SM-PKE scheme of a directly reproducible and IND-atk secure encryption scheme is also secure in the IND-atk sense. More precisely, for any PPT adversary A against the m-SM-PKE there is a PPT adversary B against the base PKE such that:*

$$\mathrm{Adv}^{\mathtt{IND-atk}}_{m-\mathtt{SM-PKE}}(A) \leq \mathrm{Adv}^{\mathtt{IND-atk}}_{\mathtt{PKE}}(B).$$

Remark. It can then be easily shown that modified ElGamal is directly reproducible (R_1 on input xP returns (sxP, s)). This eliminates the factor m in the proof of Theorem 5.5.

The above notion of reproducibility, not only permits deriving tighter security reductions, but also gives rise to a new test for detecting additional schemes which allow randomness-reuse. In fact, we will show that a modified version of the escrow ElGamal encryption scheme is directly but not weakly reproducible. Furthermore it can be easily checked that, unlike the weak and full reproducibility tests, direct reproducibility of a PKE scheme is preserved under the Fujisaki–Okamoto transformation (see Section 2.4.2). It therefore establishes a new set of CCA secure single-message multi-recipient schemes with tight security reductions in the random oracle model.

5.4.2 Modified Escrow ElGamal Encryption Scheme

Let us consider a modified version of the escrow ElGamal encryption scheme. In this scheme, which does *not* permit escrow decryption, the domain parameter is $I := (p, P, Q)$ where $Q = tP$ for $t \leftarrow \mathbb{Z}_p^*$. The key generation algorithm outputs $(1/x, xP)$ as the secret/public key-pair for $x \leftarrow \mathbb{Z}_p^*$. The encryption algorithm on input a message $\mathtt{m}$ and a public key xP returns $(U, V) := (r(xP), \mathtt{m} \cdot e(P, Q)^r)$ where r is random in $\mathbb{Z}_p^*$. One is able to decrypt this ciphertext using the secret key $1/x$ by computing $\mathtt{m} := V/e(U, (1/x)Q)$. Here $e : G \times G \rightarrow G_T$ is a non-degenerate efficiently computable bilinear map as described in Section 2.2.2.

The randomness-reuse properties of this scheme are as follows.

Theorem 5.7. *The modified escrow ElGamal encryption scheme given above is: (1) IND-CPA secure under the decisional bilinear Diffie–Hellman assumption; (2) directly reproducible; and (3) not weakly reproducible if the computational Diffie–Hellman assumption holds in G.*

Proof. (1) The security proof is analogous to that of escrow ElGamal [23].

(2) The direct reproducibility algorithm $R = (R_1, R_2, R_3)$ operates as follows. Algorithm R_1 on input a public key xP returns $(s(xP), s)$ where $\mathrm{st} := s$ is a random

element in $\mathbb{Z}_p^*$. The algorithm R_2 on input a ciphertext $(U,V) = ((xr)P, \mathtt{m} \cdot e(P,Q)^r)$ and state information st returns (sU,V). It is easily seen that R produces a valid encryption of $\mathtt{m}$ under $s(xP)$. Algorithm R_3 returns $((1/s)U,V)$. Note that the public key $s(xP)$ is identically distributed to public keys returned by the key generation algorithm.

(3) Let $(P,aP,bP) \in G^3$ denote the CDH problem instance. Our goal is to compute abP. The reproduction algorithm on input

$$(P,Q,xP,rxP,\mathtt{m} \cdot e(P,Q)^r,\mathtt{m},yP,1/y)$$

outputs $(ryP,\mathtt{m} \cdot e(P,Q)^r)$. To compute abP we pass to R the input

$$(aP,P,P,bP,e(bP,aP),1,aP,1).$$

This could be written as:

$$(P',(1/a)P',(1/a)P',(b/a)P',e(P',(1/a)P')^b,1,P',1),$$

where $P' = aP$. Note again that since R succeeds with probability 1, it will run correctly on the above input instance. Here, implicitly we have $x = 1/a$. From $rx = b/a$ we get $r = b$ and $\mathtt{m} = 1$. Note also that $y = 1$. Therefore the first component of the output of R will be $(b \cdot 1)P' = abP$. □

The direct reproducibility test also poses an interesting problem, which concerns public-key encryption schemes with chosen ciphertext security in the standard model. In particular, the construction of dREP algorithms for Cramer–Shoup type encryption schemes remains open. Alternatively, one might be able to design a new reproducibility test which admits encryption schemes which are IND-CCA secure in the standard model with tight reduction.

5.5 Extensions to the Identity-Based Setting

5.5.1 Reproducibility Notions

The notions of reproducibility considered in the previous sections can be extended to the identity-based setting in a natural way. However, due to the two level nature of IBEs, consisting of a trusted authority and users, randomness-reuse and reproducibility are better conceived in two dimensions: (1) for a constant identity, moving across different TAs; and (2) for different identities under the same TA. The latter case is, however, of greater importance in practice. For instance, similarly to the results of Bellare et al. [16], one could interpret randomness-reuse in the stateful identity-based encryption setting.

Definition 5.6. *An IBE scheme is called* fully ID-reproducible *if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1 *for any two identities* $\mathtt{ID}, \mathtt{ID}' \in \{0,1\}^*$.

1. $(\mathtt{Mpk}, \mathtt{Msk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(I)$
2. $r \leftarrow \mathcal{R}_{\mathtt{IBE}}(I)$; $\mathtt{m}, \mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{IBE}}(I)$
3. $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$; $\mathtt{c}' \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}', \mathtt{ID}', \mathtt{Mpk}; r)$
4. $D' \leftarrow \mathbb{X}_{\mathtt{IBE}}(\mathtt{ID}', \mathtt{Msk})$
5. *If* $R(\mathtt{Mpk}, \mathtt{ID}, \mathtt{c}, \mathtt{m}', \mathtt{ID}', D') = \mathtt{c}'$ *return* 1 *else return* 0

An IBE scheme is called fully TA-reproducible *if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1 *for any identity* $\mathtt{ID} \in \{0,1\}^*$.

1. $(\mathtt{Msk}, \mathtt{Mpk}), (\mathtt{Msk}', \mathtt{Mpk}') \leftarrow \mathbb{G}_{\mathtt{IBE}}(I)$
2. $r \leftarrow \mathcal{R}_{\mathtt{IBE}}(I)$; $\mathtt{m}, \mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{IBE}}(I)$

3. $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}; r)$; $\mathtt{c}' \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}', \mathtt{ID}, \mathtt{Mpk}'; r)$

4. *If* $R(\mathtt{Mpk}, \mathtt{ID}, \mathtt{c}, \mathtt{m}', \mathtt{Mpk}', \mathtt{Msk}') = \mathtt{c}'$ *return* 1 *else return* 0

One can then prove analogous reproducibility theorems as in the public-key setting. The only difference in the proofs is the presence of an extraction oracle for each TA. These are handled easily using the extraction oracle for the base IBE or the knowledge of Msk's. Note also that for full randomness-reuse across different TAs and different IDs, we need reproducibility for both IDs and TAs.

Similarly, one can formulate weak reproducibility and theorem in the IBE setting.

Definition 5.7. *An IBE scheme is called* weakly ID-reproducible *if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1 *for any three identities* $\mathtt{ID}_1, \mathtt{ID}_2, \mathtt{ID}_3 \in \{0,1\}^*$.

1. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathbb{G}_{\mathtt{IBE}}(I)$

2. $r \leftarrow \mathcal{R}_{\mathtt{IBE}}(I)$; $\mathtt{m}, \mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{IBE}}(I)$

3. $\mathtt{c}_1 \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}_1, \mathtt{Mpk}; r)$; $\mathtt{c}_2 \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}_2, \mathtt{Mpk}; r)$

4. $D_2 \leftarrow \mathbb{X}_{\mathtt{IBE}}(\mathtt{ID}_2, \mathtt{Msk})$; $D_3 \leftarrow \mathbb{X}_{\mathtt{IBE}}(\mathtt{ID}_3, \mathtt{Msk})$

5. *If* $R(\mathtt{Mpk}, \mathtt{ID}_1, \mathtt{c}_1, \mathtt{m}, \mathtt{ID}_2, D_2) \neq \mathtt{c}_2$ *return* 0

6. *If* $R(\mathtt{Mpk}, \mathtt{ID}_1, \mathtt{c}_1, \mathtt{m}', \mathtt{ID}_3, D_3) \neq R(\mathtt{Mpk}, \mathtt{ID}_2, \mathtt{c}_2, \mathtt{m}', \mathtt{ID}_3, D_3)$ *return* 0 *else return* 1

An IBE scheme is called weakly TA-reproducible *if there exists a PPT machine R such that the following experiment returns* 1 *with probability* 1 *for any identity* $\mathtt{ID} \in \{0,1\}^*$.

1. $(\mathtt{Msk}_1, \mathtt{Mpk}_1), (\mathtt{Msk}_2, \mathtt{Mpk}_2), (\mathtt{Msk}_3, \mathtt{Mpk}_3) \leftarrow \mathbb{G}_{\mathtt{IBE}}(I)$

2. $r \leftarrow \mathcal{R}_{\mathtt{IBE}}(I)$; $\mathtt{m}, \mathtt{m}' \leftarrow \mathcal{M}_{\mathtt{IBE}}(I)$

3. $\mathtt{c}_1 \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}_1; r)$; $\mathtt{c}_2 \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}, \mathtt{ID}, \mathtt{Mpk}_2; r)$

4. *If* $R(\mathtt{Mpk}_1, \mathtt{ID}, \mathtt{c}_1, \mathtt{m}, \mathtt{Mpk}_2, \mathtt{Msk}_2) \neq \mathtt{c}_2$ *return* 0

5. *If* $R(\mathtt{Mpk}_1, \mathtt{ID}, \mathtt{c}_1, \mathtt{m}', \mathtt{Mpk}_3, \mathtt{Msk}_3) \neq R(\mathtt{Mpk}_2, \mathtt{ID}, \mathtt{c}_2, \mathtt{m}', \mathtt{Mpk}_3, \mathtt{Msk}_3)$ *return* 0 *else return* 1

The direct reproducibility test can also be formulated in the identity-based setting. We omit the details here, and only mention that the definition should (1) assert the existence of an algorithm for simulation of extraction oracles under different TAs[3]; and (2) pass identities to reproducibility algorithms as they are chosen by the adversary in the security games.

5.5.2 Boneh and Franklin's Scheme

The IBE scheme of Boneh and Franklin allows for full randomness-reuse. The workflow scheme in [3] and the hidden credential systems in [27, 53] exploit this property to gain efficiency. Note that scheme considered below is only IND-CPA secure.

Theorem 5.8. *Boneh and Franklin's IBE scheme is fully ID- and TA-reproducible.*

Proof. We prove this theorem simultaneously for IDs and TAs by considering $\mathtt{aux} := \mathbb{X}_{\mathtt{IBE}}(\mathtt{ID}', \mathtt{Msk}')$ for different TAs *and* identities $\mathtt{ID}'$. The reproduction algorithm on input $(\mathtt{Mpk}, \mathtt{ID}, \mathtt{c}, \mathtt{m}', \mathtt{Mpk}', \mathtt{ID}', \mathtt{aux})$ returns

$$(\mathtt{c}_1, \mathtt{m}' \oplus H(e(\mathtt{c}_1, \mathtt{aux}))),$$

where $\mathtt{c} = (\mathtt{c}_1, \mathtt{c}_2) = (rP, \mathtt{m} \oplus H(e(\mathtt{Mpk}, H_1(\mathtt{ID}))^r))$. The correctness of this algorithm can be checked easily. □

[3] This might result in a non-polynomial-time game definition.

5.5.3 Sakai and Kasahara's Scheme

The IBE scheme of Sakai and Kasahara is an interesting case to study. We show in Lemma 5.4 that Sakai–Kasahara is not weakly reproducible for IDs or TAs. This scheme is in fact insecure with randomness-reuse across different IDs. Despite this, the scheme does allow for randomness-reuse across different TAs for a single ID. However, the reduction presented below is not a hybrid argument, and is based on an assumption which is stronger than the security of Sakai and Kasahara IBE scheme in the single-user setting [30]. This highlights the fact that a scheme which does not satisfy the reproducibility tests is not necessarily insecure with randomness-reuse.

This scheme which encrypts a message for an identity $\mathtt{ID}$ under m different TAs is defined as follows. Key generation and secret key extraction are the same as those in the Sakai–Kasahara IBE scheme as described in Section 2.4.6. Encryption and decryption algorithms are given below.

$\mathbb{E}_{m-\mathtt{TA}-\mathtt{SM}-\mathtt{IBE}}(\mathtt{m},\mathtt{ID},\mathtt{Mpk}_1,\ldots,\mathtt{Mpk}_m)$

- $r \leftarrow \mathbb{Z}_p^*$
- For $i = 1,\ldots,m$
  $U_i \leftarrow r(\mathtt{Mpk}_i + H_1(\mathtt{ID})P_1)$
- $V \leftarrow \mathtt{m} \oplus H_2(e(P_1,P_2)^r)$
- $\mathtt{c}_i \leftarrow (U_i,V)$
- $\mathbf{c} \leftarrow (\mathtt{c}_1,\ldots,\mathtt{c}_m)$
- Return $\mathbf{c}$

$\mathbb{D}_{m-\mathtt{TA}-\mathtt{SM}-\mathtt{IBE}}(\mathtt{c},S)$

- $(U,V) \leftarrow \mathtt{c}$
- $t \leftarrow e(U,S)$
- $\mathtt{m} \leftarrow V \oplus H_2(t)$
- Return $\mathtt{m}$

Theorem 5.9. *The above scheme is IND-CPA under the $(mq+1)$-BDHI assumption (see Section 2.2.2) in the random oracle model. Here q is the maximum number of extraction queries under a TA.*

Proof. (*Sketch*) The proof is an extension of that in [30]. Let A be an adversary against the scheme, we construct B attacking the $(mq+1)$-BDHI problem. Algorithm B receives the tuple $(R, xR, \ldots, x^{mq+1}R)$ as its challenge. It selects $h_0, \ldots, h_q$ and $r_1, \ldots, r_m$ randomly from $\mathbb{Z}_p^*$. It then sets $P = \prod_{j=1}^{q}[(r_1x+h_j)\ldots(r_mx+h_j)]R = \sum_{j=0}^{mq} c_j x^j R =: f(x)R$. The point P is the global group generator. Algorithm B now defines $\mathtt{Mpk}_i = r_i(xP) - h_0P$, and therefore, implicitly, $\mathtt{Msk}_i = r_ix - h_0$. Note that xP is calculable.

The non-repeat H_1 queries are answered by $h_i + h_0$ for non-challenge identities, and by h_0 for the challenge identity. Extraction queries $(\mathtt{ID}_i, \mathtt{Mpk}_j)$ are answered by computing the above product with the term $(r_jx + h_i)$ omitted. The H_2 queries are answered randomly.

The challenge is constructed by selecting a random V and setting $U_i = r_iwP$ where w is randomly chosen in $\mathbb{Z}_p^*$. The challenge tuple is well-formed:

$$U_i = s(\mathtt{Mpk}_i + h_0P) = s(r_ixP - h_0P + h_0P) = s(r_ixP) = r_i(sxP) = r_iwP,$$

where s is the implicit randomness and it equals w/x.

Now we choose a random element t from the second hash list and compute $(t/t')^{1/c_0}$, where $t' = e((f(x)-c_0)/xR, (f(x)+c_0)R)$. Note that $c_0 \neq 0$. This is the correct answer if $t = e(R,R)^s$. This value of t will be on the list as H_2, being a random oracle, one-time pads $\mathtt{m}$. □

The non-reproducibility of the IBE of scheme of Sakai and Kasahara is proved next.

Lemma 5.4. *If the IBE scheme of Sakai and Kasahara is OW-CPA secure then it is not weakly ID- or TA-reproducible.*

Proof. It is easy to see that the one-wayness of the scheme implies the hardness of the CDH problem. So let (P, aP, bP) be a CDH input instance. For weak TA-

reproducibility note that a reproduction algorithm R, when $\mathtt{m}' = \mathtt{m}$, on input

$$(I_1, \ldots, I_7) = (P, \mathtt{Mpk}, r(\mathtt{Mpk} + H_1(\mathtt{ID})P), V, \mathtt{ID}, \mathtt{Mpk}', \mathtt{Msk}')$$

outputs as its first component

$$r(\mathtt{Mpk}' + H_1(\mathtt{ID})P) = \frac{I_3}{\mathrm{dlog}_{I_1}(I_2) + H_1(I_5)}(I_7 + H_1(I_5)).$$

If the input $(aP, P + hP - h(aP), bP, V, \mathtt{ID}, aP, 1)$ where $h = H_1(\mathtt{ID})$ is passed to the reproducibility algorithm, we obtain

$$\frac{I_3}{\mathrm{dlog}_{I_1}(I_2) + H_1(I_5)}(I_7 + H_1(I_5)) = \frac{bP}{1/a + h/a - h + h}(1 + h) = abP.$$

Note that the input to the reproducibility algorithm is well-formed and R will run correctly on it.

For weak ID-reproducibility, R on input

$$(P, xP, r(xP + H_1(\mathtt{ID})P), V, \mathtt{ID}, \mathtt{ID}', 1/(x + H_1(\mathtt{ID}'))P)$$

outputs $r(xP + H_1(\mathtt{ID}')P)$ as its first component. We may now unmask $\mathtt{m}$ from V by computing

$$rP = \frac{r(xP + H_1(\mathtt{ID})P) - r(xP + H_1(\mathtt{ID}')P)}{H_1(\mathtt{ID}) - H_1(\mathtt{ID}')},$$

as $H_1(\mathtt{ID}) \neq H_1(\mathtt{ID}')$. This contradicts the one-wayness of the scheme. □

5.6 Open Problems

We conclude this chapter by presenting some problems for further research.

1. Prove the security of the modified Cramer–Shoup scheme in Section 5.2.2

and the *m*-SK-KEM in Section 5.3.2. Can the latter be done by formulating a useful notion of reproducibility for KEMs?

2. All schemes considered so far are ElGamal-based. Are there non-ElGamal-based randomness reusing schemes?

3. It remains an important open problem to construct IBE schemes in the standard model which allow for randomness-reuse across users.

4. Randomness-reuse can be studied for other cryptographic primitives such as CLE schemes, signcryption schemes or hierarchical IBE schemes. One can also consider the problem of reusing randomness across different cryptographic primitives. If achievable, constructions which are based on such primitives may be improved.

5. Are Cramer–Shoup based encryption schemes directly reproducible?

6. Are the full reproducibility and/or weak reproducibility tests necessary?

7. As we shall see in Chapter 8, it is possible to *amplify* security of a specific scheme by reusing randomness. Can this be done in a generic way?

8. Is it possible to generalise tight security reductions? More generally, what is the relation between randomness-reuse and the random self-reducibility of hard problems underlying a scheme?

Chapter 6

Cryptographic Workflow

In this chapter, following the work of Al-Riyami et al. [3], we define the notion of key encapsulation mechanism supporting cryptographic workflow (WF-KEM) and prove a composition theorem which extends the notion of hybrid encryption to cryptographic workflow. We then generically construct a WF-KEM from an identity-based encryption scheme and a secret sharing scheme. Chosen ciphertext security is achieved using one-time signatures. Adding a public-key encryption scheme we are able to modify the construction to obtain escrow-freeness. In contrast to the previous construction, we prove our generic scheme to be secure in the standard model. We refer the reader to Chapter 1 for an introduction on cryptographic workflow. This chapter first appeared in [11] and represents joint work with Manuel Barbosa.

6.1 KEM Primitives for Cryptographic Workflow

6.1.1 Access Structures, Policies and Credentials

We first explain how we treat access structures (see Section 2.3.8) in our constructions. We follow an approach similar to that in [3], but we briefly clarify this point

stating our assumptions on their meaning in real life.

Suppose that we would like to encrypt a message such that only British nationals can read. One way to achieve this would be to have a TA who issues *credentials* only to those who possess British nationality. For example, the Home Office would be the obvious TA to issue British Nationality certificates. However, it could be the case that two or more TAs are able to issue such a credential. For instance, the user's employer could, after checking the appropriate documentation, grant her a similar credential. We therefore need to specify precisely which authority we are trusting. The need for this is even more apparent when the policy is more complex. Consider the policy $\texttt{English} \wedge \texttt{English} \wedge \texttt{Adult}$, where the first two policy terms refer to nationality and language with credentials issued by the Home Office and the British Council, respectively. It could also be the case that the same authority issues credentials on age and nationality: it is up to the authority to interpret the semantics.

For this reason, we view a policy term as a pair $(\texttt{ID}, \texttt{Mpk})$ where $\texttt{ID} \in \{0,1\}^*$ is an identifier for the policy term and $\texttt{Mpk}$ is the public key of the authority issuing the credential described in $\texttt{ID}$. We denote by m the number of distinct TAs present in the system, by n the number of distinct policy terms in an access structure and by k the number of distinct policy terms in a qualifying set.

6.1.2 KEMs Supporting Cryptographic Workflow

A key encapsulation mechanism supporting cryptographic workflow (WF-KEM) is defined as a four-tuple of PPT algorithms as follows:

- $\mathbb{G}_{\texttt{WF}-\texttt{KEM}}(1^\kappa, m)$: This is the probabilistic authority key generation algorithm which on input a security parameter 1^κ and the number of TAs m, which is required to be polynomial in κ, outputs m authority secret/public key pairs $((\texttt{Msk}_i, \texttt{Mpk}_i))_{i=1}^m$, as well as a domain parameter I and descriptions of the

key, randomness and ciphertext spaces. These are denoted by $\mathcal{K}_{\mathtt{WF-KEM}}(I)$, $\mathcal{R}_{\mathtt{WF-KEM}}(I)$ and $\mathcal{C}_{\mathtt{WF-KEM}}(I)$, respectively.

- $\mathbb{X}_{\mathtt{WF-KEM}}(X, \mathtt{Msk})$: This is the probabilistic credential extraction algorithm which on input a policy term X, consisting of a policy identifier/authority public key pair $(\mathtt{ID}, \mathtt{Mpk})$, and the secret key $\mathtt{Msk}$ corresponding to $\mathtt{Mpk}$, outputs a pair $(\mathtt{crd}, X)$ which we call a *credential*.

- $\mathbb{E}_{\mathtt{WF-KEM}}(\mathcal{P})$: This is the probabilistic key encapsulation algorithm which on input an access structure $\mathcal{P}$ on n policy terms $P = \{X_1, \ldots, X_n\}$ outputs a pair $(\mathtt{k}, \mathbf{c})$ where $\mathtt{k} \in \mathcal{K}_{\mathtt{WF-KEM}}(I)$ and $\mathbf{c}$ is an encapsulation of $\mathtt{k}$.

- $\mathbb{D}_{\mathtt{WF-KEM}}(\mathbf{c}, \mathbf{crd})$: This is the deterministic decapsulation algorithm which on input an encapsulation $\mathbf{c}$ and a list of credentials $\mathbf{crd}$, outputs a key $\mathtt{k}$ or a failure symbol $\perp$.

A WF-KEM is called *correct* if for every policy $\mathcal{P}$ on n policy terms the following experiment returns 1 with probability 1. In what follows, $m, n \in \mathbb{N}$ and are polynomial in the security parameter.

1. $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{WF-KEM}}(1^{\kappa}, m)$
2. $(\mathtt{k}, \mathbf{c}) \leftarrow \mathbb{E}_{\mathtt{WF-KEM}}(\mathcal{P})$
3. $Q \leftarrow \mathcal{P}$
4. Parse $(X_{i_1}, \ldots, X_{i_k}) \leftarrow Q$
5. $[\mathbf{crd}]_j \leftarrow \mathbb{X}_{\mathtt{WF-KEM}}(X_{i_j}, \mathtt{Msk}_{i_j}), 1 \leq j \leq k$
6. If $\mathtt{k} = \mathbb{D}_{\mathtt{WF-KEM}}(\mathbf{c}, \mathbf{crd})$ return 1 else return 0

The indistinguishability games against chosen credential and ciphertext attacks for a WF-KEM are defined as follows. As in [3] we call this notion *recipient security*.

(m,n)-IND-atk

1. $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{WF-KEM}}(1^\kappa, m)$
2. $(\mathcal{P}^*, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk}_1, \ldots, \mathtt{Mpk}_m)$
3. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{WF-KEM}}(I)$
4. $(\mathtt{k}_1, \mathbf{c}^*) \leftarrow \mathbb{E}_{\mathtt{WF-KEM}}(\mathcal{P}^*)$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b, \mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{(m,n)-\mathtt{IND-atk}}_{\mathtt{WF-KEM}}(A) := |2\Pr[b' = b] - 1|.$$

Here $\mathcal{P}^*$ must be on n policy terms; O_1 and O_2 contain credential extraction and decapsulation oracles subject to the following restrictions:

- The set of queries that the adversary makes to the credential extraction oracle must not form a qualifying set of $\mathcal{P}^*$.
- The adversary cannot query the decapsulation oracle on $\mathbf{c}^*$.

We distinguish adaptive ($\mathtt{atk} = \mathtt{CCCA}$) and non-adaptive ($\mathtt{atk} = \mathtt{CCCA}^-$) chosen ciphertext and credential attacks. The difference is that in non-adaptive attacks, the adversary is not allowed to query the extraction oracle on any $X = (\mathtt{ID}, \mathtt{Mpk})$ with $X \in P^*$ in the second stage of the game (as opposed to a qualifying set of X's in the adaptive attack). Recall from Section 2.3.8 that $\mathcal{P}^*$ is a collection of susets of P^*.

As usual, a WF-KEM is called IND-CCCA (IND-CCCA$^-$) secure if all PPT attackers have negligible advantage in the above game as a function of the security parameter.

Note that WF-KEMs are intrinsically multi-user, as anyone who is able to obtain a qualifying set of credentials will be capable of decapsulating. However, in most practical cases this probably will not be the case, as the credential policy term

semantics will include the intended recipient's identity. This is related to another important characteristic of WF-KEMs. Any colluding set of TAs who can produce a qualifying set of credentials are also able to invert the encapsulation, and this means that a WF-KEM is not escrow-free.

6.1.3 KEMs Supporting Escrow-Free Cryptographic Workflow

The notion of a KEM supporting escrow-free cryptographic workflow (EFWF-KEM) implies modifying the previous primitive to remove recipient ambiguity. We follow an approach similar to [3] and [5] whereby the primitive is extended to include a recipient public and private key pair.

EFWF-KEMs are defined through five PPT algorithms. Four of these algorithms are analogous to those defined for WF-KEMs. In addition to these we add an extra user key generation algorithm:

- $\mathbb{G}_{\mathtt{EFWF-KEM}}(1^\kappa, m)$: This is the probabilistic authority key generation algorithm which on input a security parameter 1^κ and the number of TAs m, which is required to be polynomial in κ, outputs m trusted authority secret/public key pairs $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^m$, as well as a domain parameter I and descriptions of the key, randomness and ciphertext spaces. These are denoted by $\mathcal{K}_{\mathtt{EFWF-KEM}}(I)$, $\mathcal{R}_{\mathtt{EFWF-KEM}}(I)$ and $\mathcal{C}_{\mathtt{EFWF-KEM}}(I)$ respectively.

- $\mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-KEM}}(1^\kappa)$: This is the probabilistic user key generation algorithm which on input of the security parameter outputs a private/public key pair $(\mathtt{SK}, \mathtt{PK})$.

- $\mathbb{X}_{\mathtt{EFWF-KEM}}(X, \mathtt{Msk})$: This is the probabilistic credential extraction algorithm which on input a policy term X, consisting of a policy identifier/authority public key pair $(\mathtt{ID}, \mathtt{Mpk})$, and the secret key $\mathtt{Msk}$ corresponding to $\mathtt{Mpk}$, outputs a pair $(\mathtt{crd}, X)$ which we call a *credential*.

- $\mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}, \mathtt{PK})$: This is the probabilistic encapsulation algorithm which on

input an access structure $\mathcal{P}$ on n policy terms $P = \{X_1, \ldots, X_n\}$ and a public key $\mathtt{PK}$, outputs a pair $(\mathtt{k}, \mathbf{c})$ where $\mathtt{k} \in \mathcal{K}_{\mathtt{EFWF-KEM}}(I)$ and $\mathbf{c}$ is an encapsulation of $\mathtt{k}$.

- $\mathbb{D}_{\mathtt{EFWF-KEM}}(\mathbf{c}, \mathbf{crd}, \mathtt{SK})$: This is the deterministic decapsulation algorithm that on input an encapsulation $\mathbf{c}$, a list of credentials $\mathbf{crd}$, and a secret key $\mathtt{SK}$ outputs a key $\mathtt{k}$ or a failure symbol $\perp$.

An EFWF-KEM is called *correct* if for every policy $\mathcal{P}$ on n policy terms the following experiment returns 1 with probability 1. In what follows, $m, n \in \mathbb{N}$ and are polynomial in the security parameter.

1. $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{EFWF-KEM}}(1^\kappa, m)$
2. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-KEM}}(1^\kappa)$
3. $(\mathtt{k}, \mathbf{c}) \leftarrow \mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}, \mathtt{PK})$
4. $Q \leftarrow \mathcal{P}$; Parse $(X_{i_1}, \ldots, X_{i_k}) \leftarrow Q$
5. $[\mathbf{crd}]_j \leftarrow \mathbb{X}_{\mathtt{EFWF-KEM}}(X_{i_j}, \mathtt{Msk}_{i_j}), 1 \leq j \leq k$
6. If $\mathtt{k} = \mathbb{D}_{\mathtt{EFWF-KEM}}(\mathbf{c}, \mathbf{crd}, \mathtt{SK})$ return 1 else return 0

Recipient security for an EFWF-KEM is defined through a game very similar to that presented for a WF-KEM. The only difference is that here the adversary is provided with a user key pair which is generated at the beginning of the game. This captures the notion that even the user who knows the private key must possess a qualifying set of credentials to decapsulate. The game is specified below on the left. Again, $\mathcal{P}^*$ must be on at most n policy terms; the O_1 and O_2 oracles are exactly as in the previous game for adaptive ($\mathtt{atk} = \mathtt{CCCA}$) and non-adaptive chosen credential attacks ($\mathtt{atk} = \mathtt{CCCA}^{-}$).

(m,n)-IND-atk

1. $(\mathtt{Msk}_i,\mathtt{Mpk}_i)_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{EFWF-KEM}}(1^\kappa)$
2. $(\mathtt{SK},\mathtt{PK}) \leftarrow \mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-KEM}}(1^\kappa)$
3. $(\mathcal{P}^*,\mathrm{st}) \leftarrow A_1^{O_1}((\mathtt{Mpk}_i)_{i=1}^{m},\mathtt{SK},\mathtt{PK})$
4. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{EFWF-KEM}}(I)$
5. $(\mathtt{k}_1,\mathbf{c}^*) \leftarrow \mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}^*,\mathtt{PK})$
6. $b \leftarrow \{0,1\}$
7. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b,\mathbf{c}^*,\mathrm{st})$

(m,n)-IND-CCA

1. $(\mathtt{Msk}_i,\mathtt{Mpk}_i)_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{EFWF-KEM}}(1^\kappa)$
2. $(\mathtt{SK},\mathtt{PK}) \leftarrow \mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-KEM}}(1^\kappa)$
3. $(\mathcal{P}^*,\mathrm{st}) \leftarrow A_1^{O_1}((\mathtt{Msk}_i,\mathtt{Mpk}_i)_{i=1}^{m},\mathtt{PK})$
4. $\mathtt{k}_0 \leftarrow \mathcal{K}_{\mathtt{EFWF-KEM}}(I)$
5. $(\mathtt{k}_1,\mathbf{c}^*) \leftarrow \mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}^*,\mathtt{PK})$
6. $b \leftarrow \{0,1\}$
7. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b,\mathbf{c}^*,\mathrm{st})$

To capture escrow-freeness, we follow the approach in [3] and define *external security* through the indistinguishability game shown above on the right. Note that the adversary controls everything except the user's secret key. Here $\mathcal{P}^*$ must be on n policy terms; O_1 and O_2 denote a decapsulation oracle subject to the restriction that the adversary cannot query it on $\mathbf{c}^*$. An EFWF-KEM is called IND-CCCA (IND-CCCA$^-$) and IND-CCA secure if all PPT attackers have negligible advantage in the above games as a function of the security parameter, where advantages are defined as

$$\mathrm{Adv}^{(m,n)-\mathtt{IND-atk}}_{\mathtt{EFWF-KEM}}(A) := |2\Pr[b'=b]-1|$$

for the recipient security and

$$\mathrm{Adv}^{(m,n)-\mathtt{IND-CCA}}_{\mathtt{EFWF-KEM}}(A) := |2\Pr[b'=b]-1|$$

for the external security. Here we require m and n to be polynomial in the security parameter.

6.1.4 Hybrid Encryption Supporting Cryptographic Workflow

The concept and security model of an encryption scheme supporting escrow-free cryptographic workflow (EFWF-ENC), as proposed in [3], are defined in a very similar manner to an EFWF-KEM. Following [3], an encryption scheme supporting escrow-free cryptographic workflow (EFWF-ENC) is defined via five PPT algorithms. The algorithms $\mathbb{G}_{\mathtt{EFWF-ENC}}(1^\kappa, m)$, $\mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-ENC}}(1^\kappa)$ and $\mathbb{X}_{\mathtt{EFWF-ENC}}(X, \mathtt{Msk})$ are identical to that of an EFWF-KEM. Encryption and decryption are as follows.

- $\mathbb{E}_{\mathtt{EFWF-ENC}}(\mathtt{m}, \mathcal{P}, \mathtt{PK})$: This is the probabilistic encryption algorithm which on input a message $\mathtt{m}$, an access structure $\mathcal{P}$ and a public key $\mathtt{PK}$, outputs a ciphertext $\mathbf{c}$.

- $\mathbb{D}_{\mathtt{EFWF-ENC}}(\mathbf{c}, \mathbf{crd}, \mathtt{SK})$: This is the deterministic decryption algorithm which on input a ciphertext $\mathbf{c}$, a list of credentials $\mathbf{crd}$ and a secret key $\mathtt{SK}$, outputs a message $\mathtt{m}$ or a failure symbol $\perp$.

The correctness of an EFWF-ENC scheme is defined similarly to that of an EFWF-KEM.

The recipient security model is defined below. Here, $\mathcal{P}^*$ must be on n policy terms, and O_1 and O_2 denote credential extraction and decryption oracles subject to the following restrictions. In the CCCA model the set of queries that the adversary makes to the credential extraction oracle should not form a qualifying subset of P^*. In the CCCA$^-$ model there is the additional restriction that the adversary cannot call the extraction oracle on any $X = (\mathtt{ID}, \mathtt{Mpk})$ with $X \in P^*$ in the second stage of the game. Also, the adversary cannot query the decryption oracle on $\mathbf{c}^*$.

(m,n)-IND-atk

1. $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{EFWF-ENC}}(1^\kappa, m)$
2. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-ENC}}(1^\kappa)$
3. $(\mathtt{m}_0, \mathtt{m}_1, \mathcal{P}^*, \mathrm{st}) \leftarrow A_1^{O_1}((\mathtt{Mpk}_i)_{i=1}^{m}, \mathtt{SK}, \mathtt{PK})$
4. $b \leftarrow \{0,1\}$
5. $\mathbf{c}^* \leftarrow \mathbb{E}_{\mathtt{EFWF-ENC}}(\mathtt{m}_b, \mathcal{P}^*, \mathtt{PK})$
6. $b' \leftarrow A_2^{O_2}(\mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{(m,n)-\mathtt{IND-atk}}_{\mathtt{EFWF-ENC}}(A) := |2\Pr[b' = b] - 1|.$$

Escrow-freeness, or external security, is captured via the game below. Here, $\mathcal{P}^*$ must be on n policy terms, and O_1 and O_2 denote a decryption oracle subject to the restriction that the adversary cannot query it on $\mathbf{c}^*$.

(m,n)-IND-CCA

1. $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^{m} \leftarrow \mathbb{G}_{\mathtt{EFWF-ENC}}(1^\kappa, m)$
2. $(\mathtt{SK}, \mathtt{PK}) \leftarrow \mathbb{G}^{\mathtt{U}}_{\mathtt{EFWF-ENC}}(1^\kappa)$
3. $(\mathtt{m}_0, \mathtt{m}_1, \mathcal{P}^*, \mathrm{st}) \leftarrow A_1^{O_1}((\mathtt{Msk}_i, \mathtt{Mpk}_i)_{i=1}^{m}, \mathtt{PK})$
4. $b \leftarrow \{0,1\}$
5. $\mathbf{c}^* \leftarrow \mathbb{E}_{\mathtt{EFWF-ENC}}(\mathtt{m}_b, \mathcal{P}^*, \mathtt{PK})$
6. $b' \leftarrow A_2^{O_2}(\mathbf{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{(m,n)-\mathtt{IND-CCA}}_{\mathtt{EFWF-ENC}}(A) := |2\Pr[b' = b] - 1|.$$

Using an EFWF-KEM and a standard DEM with compatible key spaces, one can construct a hybrid encryption scheme supporting escrow-free cryptographic workflow in the usual way:

$\mathbb{E}_{\mathtt{EFWF-ENC}}(\mathtt{m}, \mathcal{P}, \mathtt{PK})$

- $(\mathtt{k}, \overline{\mathbf{c}}) \leftarrow \mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}, \mathtt{PK})$
- $\mathtt{c} \leftarrow \mathbb{E}_{\mathtt{DEM}}(\mathtt{m}, \mathtt{k})$
- $\mathbf{c} \leftarrow (\overline{\mathbf{c}}, \mathtt{c})$
- Return $\mathbf{c}$

$\mathbb{D}_{\mathtt{EFWF-ENC}}(\mathbf{c}, \mathbf{crd}, \mathtt{SK})$

- $(\overline{\mathbf{c}}, \mathtt{c}) \leftarrow \mathbf{c}$
- $\mathtt{k} \leftarrow \mathbb{D}_{\mathtt{EFWF-KEM}}(\overline{\mathbf{c}}, \mathbf{crd}, \mathtt{SK})$
- If $\mathtt{k} = \perp$ return $\perp$
- $\mathtt{m} \leftarrow \mathbb{D}_{\mathtt{DEM}}(\mathtt{c}, \mathtt{k})$
- Return $\mathtt{m}$

The following theorem relates the security of this hybrid encryption scheme to that of its EFWF-KEM and DEM components. A similar result holds for non-escrow free primitives.

Theorem 6.1. *The hybrid EFWF-ENC scheme as constructed above is secure in the recipient and external security models if the underlying EFWF-KEM and DEM are secure. More precisely, for* $\mathtt{atk} \in \{\mathtt{CCCA}, \mathtt{CCCA}^{-}\}$*, and any PPT adversary A against the EFWF-ENC scheme, there are PPT adversaries* B_1 *and* B_2 *against the EFWF-KEM and the DEM such that:*

$$\mathrm{Adv}^{(m,n)-\mathtt{IND}-\mathtt{atk}}_{\mathtt{EFWF}-\mathtt{ENC}}(A) \leq 2\mathrm{Adv}^{(m,n)-\mathtt{IND}-\mathtt{atk}}_{\mathtt{EFWF}-\mathtt{KEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG}-\mathtt{CCA}}_{\mathtt{DEM}}(B_2),$$

$$\mathrm{Adv}^{(m,n)-\mathtt{IND}-\mathtt{CCA}}_{\mathtt{EFWF}-\mathtt{ENC}}(A) \leq 2\mathrm{Adv}^{(m,n)-\mathtt{IND}-\mathtt{CCA}}_{\mathtt{EFWF}-\mathtt{KEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG}-\mathtt{CCA}}_{\mathtt{DEM}}(B_2).$$

The proof of this theorem is similar to that of composition of *m*-SK-IBKEMs and DEMs presented in Theorem 4.1.

6.2 Generic Constructions

6.2.1 A WF-KEM Construction

We first present a construction of a WF-KEM using an IBE scheme, a secret sharing scheme and a one-time signature scheme (see Sections 2.3.8 and 2.3.7 for defini-

tions).

The authority key generation and credential extraction algorithms of the resulting WF-KEM are direct adaptations of the master key generation and secret key extraction algorithms of the underlying IBE:

- $\mathbb{G}_{\mathtt{WF-KEM}}(1^\kappa, m)$: Runs the $\mathbb{G}_{\mathtt{IBE}}(1^\kappa)$ algorithm m times in order to obtain $((\mathtt{Msk}_i, \mathtt{Mpk}_i))_{i=1}^m$. The key space is $\mathcal{K}_{\mathtt{WF-KEM}}(\kappa) = \{0,1\}^\kappa$.
- $\mathbb{X}_{\mathtt{WF-KEM}}(X, \mathtt{Msk})$: Parses X to get $(\mathtt{ID}, \mathtt{Mpk})$, extracts $\mathtt{crd} = \mathbb{X}_{\mathtt{IBE}}(\mathtt{ID}, \mathtt{Msk})$ and returns $(\mathtt{crd}, X)$.

We let $(\mathbb{G}_{\mathtt{OTS}}, \mathtt{Sig}, \mathtt{Ver})$ denote a one-time signature scheme and let $(\mathbb{S}, \mathbb{S}^{-1})$ be a secret sharing scheme. The encapsulation and decapsulation algorithms are as follows.

$\mathbb{E}_{\mathtt{WF-KEM}}(\mathcal{P})$

- $(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathbb{G}_{\mathtt{OTS}}(1^\kappa)$
- $\mathtt{k} \leftarrow \mathcal{K}_{\mathtt{WF-KEM}}(\kappa)$
- $(\mathbf{shr}, \mathtt{aux}) \leftarrow \mathbb{S}(1^\kappa, \mathtt{k}, \mathcal{P})$
- For $j = 1, \ldots, n$
  $(\mathtt{ID}, \mathtt{Mpk}) \leftarrow X_j$
  $\mathtt{c}_j \leftarrow \mathbb{E}_{\mathtt{IBE}}([\mathbf{shr}]_j || \mathtt{vk}, \mathtt{ID}, \mathtt{Mpk})$
- $\mathbf{c} \leftarrow (\mathtt{c}_1, \ldots, \mathtt{c}_n, \mathtt{vk}, \mathtt{aux}, \mathcal{P})$
- $\sigma \leftarrow \mathtt{Sig}(\mathbf{c}, \mathtt{sk})$
- Return $(\mathtt{k}, \mathbf{c} || \sigma)$

$\mathbb{D}_{\mathtt{WF-KEM}}(\mathbf{c} || \sigma, \mathbf{crd})$

- If $\mathtt{Ver}(\mathbf{c}, \sigma, \mathtt{vk}) \neq 1$ return $\perp$
- $(\mathtt{c}_1, \ldots, \mathtt{c}_n, \mathtt{vk}, \mathtt{aux}, \mathcal{P}) \leftarrow \mathbf{c}$
- For $j = 1, \ldots, k$
  $(\mathtt{crd}, X) \leftarrow [\mathbf{crd}]_j$
  Find $\mathtt{c}_i$ corresponding to X
  $([\mathbf{shr}]_j || \mathtt{vk}_j) \leftarrow \mathbb{D}_{\mathtt{IBE}}(\mathtt{c}_i, \mathtt{crd})$
  If $([\mathbf{shr}]_j || \mathtt{vk}_j) = \perp$ return $\perp$
  If $\mathtt{vk}_j \neq \mathtt{vk}$ return $\perp$
- $\mathtt{k} \leftarrow \mathbb{S}^{-1}(\mathbf{shr}, \mathtt{aux})$
- If $\mathtt{k} = \perp$ return $\perp$
- Return $\mathtt{k}$

Note that, similarly to what is done in [43] for multiple encryption in the public-key setting, one could use an IBE primitive modified to include non-malleable

public labels to bind $\mathtt{vk}$ to each individual $\mathtt{c}_j$. We chose not to do this so that we could base our construction on the more standard IBE primitive and security model. The security of the above construction is captured via the following theorem.

Theorem 6.2. *The above construction is (m,n)-IND-CCCA secure if the underlying IBE is IND-CCA secure, the OTS is UF secure, and the secret sharing scheme is information-theoretically secure. More precisely, for any PPT adversary A against the WF-KEM, there are PPT adversaries B_1 and B_2 against the OTS and the IBE scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCCA}}_{\mathtt{WF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2mn^2 \cdot \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(B_2).$$

Proof. We prove the theorem using a sequence of five games $\mathtt{Game}_0, \ldots, \mathtt{Game}_4$. Let A be an adversary against the generic workflow construction. We denote by S_i the event that A guesses the challenge bit correctly in $\mathtt{Game}_i$.

Let $\mathtt{Game}_0$ be the original IND-CCCA attack game. Hence

$$(1/2) \cdot \mathrm{Adv}^{\mathtt{IND-CCCA}}_{\mathtt{WF-KEM}}(A) = |\Pr[S_0] - 1/2|.$$

To obtain $\mathtt{Game}_1$ we introduce a single change: all decapsulation queries where the OTS verification key included in the challenge is reused by the adversary are answered immediately with $\perp$.

We claim that A's probability of success changes negligibly. Let E denote the event that the adversary submits for decapsulation a valid ciphertext $(\mathbf{c}||\sigma)$, different from the challenge ciphertext $(\mathbf{c}^*||\sigma^*)$[1]. Given that $\mathtt{Game}_0$ and $\mathtt{Game}_1$ are

[1]Note that *different* in this case means a single bit and, in particular, allows the attacker to reuse $\mathbf{c}^*$ or σ^*.

identical, unless E occurs, we have

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[E].$$

To show that this difference is negligible, it suffices to demonstrate that $\Pr[E]$ must be negligible. This follows easily from the observation that any adversary that causes E to occur with non-negligible probability can be used to directly construct an algorithm B_1 which wins the strong unforgeability game against the OTS scheme with advantage $\Pr[E]$. Therefore

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1).$$

Now we change $\mathtt{Game}_1$ so that decapsulation queries immediately return $\perp$ for all ciphertexts which reuse one or more challenge components $\mathtt{c}_i^*$ and corresponding policy terms in $\mathcal{P}^*$. We call this new game $\mathtt{Game}_2$ and we claim that

$$\Pr[S_1] = \Pr[S_2].$$

To prove this claim, we argue that $\perp$ is the correct decapsulation result for this type of ciphertext in $\mathtt{Game}_1$. To see this, note that all queries which reuse $\mathtt{vk}^*$ are already returning $\perp$, so we only need to consider the case where $\mathtt{vk} \neq \mathtt{vk}^*$. However, by construction, all $\mathtt{c}_i^*$ components will return $\mathtt{vk}^*$ when decrypted, causing the decapsulation consistency check to fail.

We obtain $\mathtt{Game}_3$ by changing the way in which the challenge encapsulation is constructed: the $\mathtt{c}_j$ challenge components are constructed using shares resulting from a completely random key $\mathtt{k}_2$ (but still using $\mathtt{aux}_1^*$).

Once again, we claim that A's probability of success changes only negligibly

when one moves to $\mathtt{Game}_3$. We prove this claim using a hybrid argument similar to that in [13].

We show that if $|\Pr[S_2] - \Pr[S_3]|$ is non-negligible, then it is possible to build an algorithm B_2 which runs A as a subroutine, interpolates between the two games, and has non-negligible advantage in the IND-CCA game that defines the security of the IBE scheme. Algorithm B_2 works as follows:

- B_2 chooses a random value ℓ in the range $1,\ldots,n$, where n is the number of policy terms.
- B_2 generates the key pairs for $m-1$ credential authorities and obtains the m-th master public key from the external IBE attack game. B_2 randomly permutes these public keys and passes them on to A.
- On input a challenge policy $\mathcal{P}^*$, algorithm B_2 constructs the challenge encapsulation as follows:
  - If the master public key from the external IBE game is not associated with the ℓ-th policy term (event F_1), B_2 terminates. Let X_ℓ be the first component of the ℓ-th policy term, and $\mathtt{ID}_\ell$ the identifier inside it.
  - B_2 chooses three keys $\mathtt{k}_0$, $\mathtt{k}_1$ and $\mathtt{k}_2$ at random, passes $\mathtt{k}_1$ and $\mathtt{k}_2$ together with $\mathcal{P}^*$ to the secret sharing algorithm to obtain $(\mathbf{shr}_1^*, \mathtt{aux}_1^*)$ and $(\mathbf{shr}_2^*, \mathtt{aux}_2^*)$.
  - B_2 runs $\mathbb{G}_{\mathtt{OTS}}$ to obtain $(\mathtt{vk}^*, \mathtt{sk}^*)$.
  - For shares $1,\ldots,\ell-1$, B_2 constructs $\mathtt{c}_j^*$ using $[\mathbf{shr}_1^*]_j||\mathtt{vk}^*$.
  - For the ℓ-th share, B_2 calls the external challenge oracle on $\mathtt{ID}_\ell$ with $(\mathtt{m}_0, \mathtt{m}_1)$, where $\mathtt{m}_0 = [\mathbf{shr}_2^*]_\ell||\mathtt{vk}^*$ and $\mathtt{m}_1 = [\mathbf{shr}_1^*]_\ell||\mathtt{vk}^*$.
  - For all remaining shares, B_2 constructs $\mathtt{c}_j^*$ using $[\mathbf{shr}_2^*]_j||\mathtt{vk}^*$.

  - B_2 now generates a random bit b and provides $\mathtt{k}_b$ to the adversary, along with the challenge ciphertext (which uses $\mathtt{aux}_1^*$).

- Credential extraction queries are handled as follows:
  - The knowledge of the master secret keys on $m-1$ of the authorities allows B_2 to directly answer most extraction queries using the IBE extraction algorithm.
  - For credentials associated with the authority from the external security game, B_2 will call the secret key extraction oracle provided in that game.
  - Algorithm B_2 will terminate if A chooses to extract the secret key associated with X_ℓ (event F_2), as this would be an invalid query in the IBE game.

- Decapsulation queries are answered as follows:
  - The necessary credentials are obtained by running the credential extraction simulation algorithm above.
  - The exception is X_ℓ in the challenge, for which B_2 simply calls the external decryption oracle. Note that by the rules in $\mathtt{Game}_2$ the adversary will not be able to force B_2 to perform a decryption query which is disallowed in the IBE security game. All queries associating $\mathtt{c}_\ell^*$ with X_ℓ are immediately answered with $\perp$.

- When A returns a bit b', algorithm B_2 will return 1 if $b = b'$ and 0 otherwise.

First we look at the probability that B_2 does not fail.

We call the event that B_2 fails `Fail`. Since the events F_1 and F_2 are independent, we have:

$$\Pr[\neg \mathtt{Fail}] = \Pr[\neg F_1 \wedge \neg F_2] = \Pr[\neg F_1] \cdot [\neg F_2] \geq \frac{1}{mn}.$$

The latter inequality follows from the following observations:

- There is a 1-in-m probability that A will output a policy in which the ℓ-th share must be extracted under the authority corresponding to the external IBE game.
- For any non-trivial policy, there is at least one credential which A cannot extract. Hence there is at least a 1-in-n chance that this is the ℓ-th share.

Let $\hat{b}$ be the bit returned by B_2 and $\bar{b}$ be the secret bit in the external IBE security game. We have

$$\begin{aligned}\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(B_2) &= |\Pr[\hat{b}=1|\bar{b}=1]-\Pr[\hat{b}=1|\bar{b}=0]| \\ &= \Pr[\neg\mathtt{Fail}]\cdot|\Pr[\hat{b}=1|\bar{b}=1\wedge\neg\mathtt{Fail}]-\Pr[\hat{b}=1|\bar{b}=0\wedge\neg\mathtt{Fail}]|.\end{aligned}$$

Let us now focus on executions of B_2 which do not abnormally terminate. It is clear that algorithm B_2 runs A in the environment of $\mathtt{Game}_2$ if $\ell = n$ and the external challenge oracle uses $\mathtt{m}_1$. Similarly, algorithm B_2 runs A in the environment of $\mathtt{Game}_3$ if $\ell = 1$ and the external challenge oracle uses $\mathtt{m}_2$. This is true regardless of the fact that the adversary will be able to open up some of the $\mathtt{c}_j$ components in the challenge which may contain shares associated with $\mathtt{k}_2$. This is guaranteed by the information-theoretical security of the secret sharing algorithm and by the fact that the adversary is never allowed to obtain a qualifying set of credentials.

Hence, we have

$$\Pr[S_2] = \Pr[\hat{b}=1|\ell=n\wedge\bar{b}=1\wedge\neg\mathtt{Fail}],$$

and

$$\Pr[S_3] = \Pr[\hat{b}=1|\ell=1\wedge\bar{b}=0\wedge\neg\mathtt{Fail}].$$

Because B_2 generates ℓ uniformly at random at the beginning of its operation,

the following summations hold for any execution of B_2:

$$\Pr[\hat{b} = 1 | \overline{b} = 1 \wedge \neg \texttt{Fail}] = \frac{1}{n} \sum_{i=1}^{n} (\Pr[\hat{b} = 1 | \ell = i \wedge \overline{b} = 1 \wedge \neg \texttt{Fail}])$$

$$\Pr[\hat{b} = 1 | \overline{b} = 0 \wedge \neg \texttt{Fail}] = \frac{1}{n} \sum_{i=1}^{n} (\Pr[\hat{b} = 1 | \ell = i \wedge \overline{b} = 0 \wedge \neg \texttt{Fail}]).$$

Now we observe that for any $2 \leq i \leq n$, by construction, B_2 guarantees the following

$$\Pr[\hat{b} = 1 | \ell = i \wedge \overline{b} = 0 \wedge \neg \texttt{Fail}] = \Pr[\hat{b} = 1 | \ell = (i-1) \wedge \overline{b} = 1 \wedge \neg \texttt{Fail}].$$

Cancelling out the summation terms, we obtain

$$\begin{aligned} \Pr[\hat{b} &= 1 | \overline{b} = 1 \wedge \neg \texttt{Fail}] - \Pr[\hat{b} = 1 | \overline{b} = 0 \wedge \neg \texttt{Fail}] = \\ &= \frac{1}{n} (\Pr[\hat{b} = 1 | \ell = n \wedge \overline{b} = 1 \wedge \neg \texttt{Fail}] - \Pr[\hat{b} = 1 | \ell = 1 \wedge \overline{b} = 0 \wedge \neg \texttt{Fail}]). \end{aligned}$$

Putting these results together, we have

$$\mathrm{Adv}^{\texttt{IND-CCA}}_{\texttt{IBE}}(B_2) = \frac{1}{n} \cdot \Pr[\neg \texttt{Fail}] \cdot |\Pr[S_2] - \Pr[S_3]|,$$

and finally

$$|\Pr[S_2] - \Pr[S_3]| \leq mn^2 \cdot \mathrm{Adv}^{\texttt{IND-CCA}}_{\texttt{IBE}}(B_2),$$

which demonstrates that the advantage of any adversary in $\texttt{Game}_3$ must be negligibly different from that in $\texttt{Game}_2$ if the underlying IBE is IND-CCA secure.

To complete the proof, we introduce a final game $\texttt{Game}_4$ where the only difference to $\texttt{Game}_3$ is the fact that we replace the $\texttt{aux}_1^*$ component in the challenge by $\texttt{aux}_2^*$. Again, the information-theoretical security of the secret sharing scheme

guarantees that A's view in the two games is identical:

$$\Pr[S_3] = \Pr[S_4].$$

Finally, we have that because no information regarding the secret bit chosen by the challenger can be leaked by the challenge in $\texttt{Game}_4$, the adversary can have no advantage:

$$\Pr[S_4] = 1/2.$$

Putting together the previous results we obtain the expression in Theorem 6.2.

□

For computational secret sharing schemes we obtain the following. This result is further explained in Section 6.3.

Theorem 6.3. *The above construction is (m,n)-IND-CCCA^{-} secure if the underlying IBE scheme is IND-CCA secure, the OTS is UF secure, and the secret sharing scheme is IND-SSA secure. More precisely, for any PPT adversary A against the WF-KEM, there are PPT adversaries B_1, B_2 and B_3 against the OTS, the IBE scheme and the computational secret sharing scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCCA}^{-}}_{\mathtt{WF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2mn^2 \cdot \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(B_2) + \mathrm{Adv}^{\mathtt{IND-SSA}}_{\mathtt{SS}}(B_3).$$

Proof. (*Sketch*) The proof is very similar to that for Theorem 6.2. However, in this case, we know exactly which credentials the adversary has extracted during the first stage, and that it is unable to extract credentials related to the challenge in stage two. This makes it possible to show that, if the IBE scheme is IND-CCA secure, the adversary's advantage changes negligibly if we change the ciphertext components to which the adversary has no access by encrypting random bit strings of appropriate length. Once in this game environment, the adversary's advantage

can then be used to directly win the IND-SSA game against the secret sharing scheme. The simulator selects the shares that the adversary will be recovering from the external IND-SSA game when it is about to construct the challenge. Since all the other ciphertext components contain random data, any advantage the adversary obtains must come from attacking the secret sharing scheme. □

6.2.2 An EFWF-KEM Construction

We now extend the previous generic construction to achieve escrow-freeness. We build an EFWF-KEM using an additional component: a PKE scheme. The authority key generation and credential extraction algorithms are as in the WF-KEM construction. The user key generation algorithm is that of the underlying PKE scheme. Finally, the encapsulation and decapsulation algorithms are:

$\mathbb{E}_{\mathtt{EFWF-KEM}}(\mathcal{P}, \mathtt{PK})$

- $(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathbb{G}_{\mathtt{OTS}}(1^\kappa)$
- $\mathtt{k}_1, \mathtt{k}_2 \leftarrow \mathcal{K}_{\mathtt{EFWF-KEM}}(\kappa)$
- $(\mathbf{shr}, \mathtt{aux}) \leftarrow \mathbb{S}(1^\kappa, \mathtt{k}_1, \mathcal{P})$
- $\overline{\mathtt{c}} \leftarrow \mathbb{E}_{\mathtt{PKE}}(\mathtt{k}_2 || \mathtt{vk}, \mathtt{PK})$
- For $j = 1, \ldots, n$
  $(\mathtt{ID}, \mathtt{Mpk}) \leftarrow X_j$
  $\mathtt{c}_j \leftarrow \mathbb{E}_{\mathtt{IBE}}([\mathbf{shr}]_j || \mathtt{vk}, \mathtt{ID}, \mathtt{Mpk})$
- $\mathbf{c} \leftarrow (\overline{\mathtt{c}}, \mathtt{c}_1, \ldots, \mathtt{c}_n, \mathtt{vk}, \mathtt{aux}, \mathcal{P})$
- $\sigma \leftarrow \mathtt{Sig}(\mathbf{c}, \mathtt{sk})$
- Return $(\mathtt{k}_1 \oplus \mathtt{k}_2, \mathbf{c} || \sigma)$

$\mathbb{D}_{\mathtt{EFWF-KEM}}(\mathbf{c} || \sigma, \mathbf{crd}, \mathtt{SK})$

- If $\mathtt{Ver}(\mathbf{c}, \sigma, \mathtt{vk}) \neq 1$ return $\perp$
- $(\overline{\mathtt{c}}, \mathtt{c}_1, \ldots, \mathtt{c}_n, \mathtt{vk}, \mathtt{aux}, \mathcal{P}) \leftarrow \mathbf{c}$
- For $j = 1, \ldots, k$
  $(\mathtt{crd}, X) \leftarrow [\mathbf{crd}]_j$
  Find $\mathtt{c}_i$ corresponding to X
  $([\mathbf{shr}]_j || \mathtt{vk}_j) \leftarrow \mathbb{D}_{\mathtt{IBE}}(\mathtt{c}_i, \mathtt{crd})$
  If $([\mathbf{shr}]_j || \mathtt{vk}_j) = \perp$ return $\perp$
  If $\mathtt{vk}_j \neq \mathtt{vk}$ return $\perp$
- $\mathtt{k}_1 \leftarrow \mathbb{S}^{-1}(\mathbf{shr}, \mathtt{aux})$
- $\mathtt{k}_2 \leftarrow \mathbb{D}_{\mathtt{PKE}}(\overline{\mathtt{c}}, \mathtt{SK})$
- If $\mathtt{k}_1 = \perp$ or $\mathtt{k}_2 = \perp$ return $\perp$
- Return $\mathtt{k}_1 \oplus \mathtt{k}_2$

Again we have two security results which depend on the security provided by the underlying secret sharing scheme.

Theorem 6.4. *The EFWF-KEM construction given above is (m,n)-IND-CCCA and (m,n)-IND-CCA secure if the underlying PKE and IBE are IND-CCA secure, the OTS is UF secure, and the secret sharing scheme is information-theoretically secure. More precisely, for any PPT adversary A against the EFWF-KEM, there are PPT adversaries B_1 and B_2 against the OTS and the IBE/PKE scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCCA}}_{\mathtt{EFWF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2mn^2 \cdot \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(B_2),$$

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{EFWF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}}(B_2).$$

We present the proof in two stages. First we address IND-CCA security, and then IND-CCCA security.

Lemma 6.1. *The construction is (m,n)-IND-CCA secure if the underlying PKE is IND-CCA secure and the OTS is UF secure.*

Proof. We construct this proof using a sequence of four games $\mathtt{Game}_0, \ldots, \mathtt{Game}_3$. Let A be an adversary against the generic workflow construction. We denote by S_i the event that A guesses the challenge bit correctly in $\mathtt{Game}_i$.

Let $\mathtt{Game}_0$ be the original IND-CCA attack game. Hence

$$(1/2) \cdot \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{EFWF-KEM}}(A) = |\Pr[S_0] - 1/2|.$$

To obtain $\mathtt{Game}_1$ we introduce a single change: all decapsulation queries where the OTS verification key included in the challenge is reused by the adversary are answered immediately with $\perp$.

We claim that A's probability of success changes negligibly. Let E denote

the event that the adversary submits for decapsulation a valid ciphertext $(\mathbf{c}||\sigma)$, different from the challenge ciphertext $(\mathbf{c}^*||\sigma^*)$.

Given that $\texttt{Game}_0$ and $\texttt{Game}_1$ are identical, unless E occurs, we have

$$|\Pr[S_0] - \Pr[S_1]| \leq \Pr[E].$$

To show that this difference is negligible, it suffices to demonstrate that $\Pr[E]$ must be negligible. This follows easily from the observation that any adversary that causes E to occur with non-negligible probability can be used to directly construct an algorithm B_1 which wins the UF game against the OTS scheme with advantage $\Pr[E]$. Therefore:

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1).$$

Now we change $\texttt{Game}_1$ so that decapsulation queries immediately return $\perp$ for all ciphertexts which reuse one or more challenge components $\mathtt{c}_i^*$ and corresponding policy terms in $\mathcal{P}^*$. We call this new game $\texttt{Game}_2$ and we claim that

$$\Pr[S_1] = \Pr[S_2].$$

To prove this claim, we argue that $\perp$ is the correct decapsulation result for this type of ciphertext in $\texttt{Game}_1$. To see this, note that all queries which reuse $\mathtt{vk}^*$ are already returning $\perp$, so we only need to consider the case where $\mathtt{vk} \neq \mathtt{vk}^*$. However, by construction, all $\mathtt{c}_i^*$ components will return $\mathtt{vk}^*$ when decrypted, causing the decapsulation consistency check to fail.

Finally, we obtain $\texttt{Game}_3$ by changing the way in which the challenge encapsulation is constructed: the PKE challenge component is constructed using a completely random string of the correct size.

Once again, we claim that A's probability of success changes only negligibly when one moves to $\texttt{Game}_3$. To prove this claim we show that if $|\Pr[S_2] - \Pr[S_3]|$ is non-negligible, then it is possible to build an algorithm B_2 which runs A as a subroutine, interpolates between the two games, and has non-negligible advantage in the IND-CCA game that defines the security of the PKE scheme. B_2 works as follows:

- B_2 generates all the IBE-related parameters itself and obtains the public key for the PKE scheme from the IND-CCA game. These are all handed over to the adversary.
- Eventually the adversary outputs a challenge policy $\mathcal{P}^*$ and B_2 constructs the challenge as follows:
  - B_2 generates the OTS key pair $(\texttt{vk}^*, \texttt{sk}^*)$.
  - B_2 generates three secret keys $\texttt{k}_0$, $\texttt{k}_1$ and $\texttt{k}_2$.
  - B_2 passes $\texttt{k}_2 || \texttt{vk}^*$ and a completely random bit string of the same size to the external IND-CCA challenge oracle.
  - B_2 uses $\texttt{k}_1$ to construct the part of the challenge ciphertext which relies on the secret sharing scheme and the IBE scheme.
  - B_2 calculates the OTS and completes the challenge ciphertext.
  - Now B_2 flips a coin b. If $b = 0$ then B_2 hands over $\texttt{k}_0$ and the challenge to the adversary. Otherwise, the adversary gets $\texttt{k}_1 \oplus \texttt{k}_2$ and the challenge ciphertext.
- Eventually the adversary will output its guess b', and B_2 will return 1 if $b = b'$ and 0 otherwise.
- Throughout its entire operation B_2 is able to answer decapsulation queries correctly according to the rules of $\texttt{Game}_2$. It takes advantage of its knowledge

of all the IBE parameters and the external PKE decryption oracle to achieve that. Again, A is not able to force B_2 into placing an invalid query to this oracle because by the rules of $\texttt{Game}_2$ any query which includes the PKE challenge ciphertext can be immediately answered with $\perp$.

Let $\hat{b}$ denote the secret bit in the PKE IND-CCA game, and $\bar{b}$ denote B_2's guess. It is clear that B_2 will run A in an environment consistent with $\texttt{Game}_2$ or $\texttt{Game}_3$, depending on whether the PKE challenge oracle encrypts $\texttt{k}_2||\texttt{vk}^*$ ($\hat{b}=0$) or the random string ($\hat{b}=1$). Hence we can write:

$$\Pr[S_2] = \Pr[b = b' | \hat{b} = 0] = \Pr[\bar{b} = 1 | \hat{b} = 0],$$

$$\Pr[S_3] = \Pr[b = b' | \hat{b} = 1] = \Pr[\bar{b} = 1 | \hat{b} = 1].$$

However, by definition, we have that

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}}(B_2) = |\Pr[\bar{b} = 1 | \hat{b} = 0] - \Pr[\bar{b} = 1 | \hat{b} = 1]|,$$

which leads to

$$|\Pr[S_2] - \Pr[S_3]| = \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}}(B_2).$$

Finally we observe that in $\texttt{Game}_3$ the adversary can have no advantage, since no information about $\texttt{k}_2$ is present in the challenge. Hence

$$\Pr[S_3] = 1/2.$$

The lemma follows from the combination of the game transition results. □

Lemma 6.2. *The construction is (m,n)-IND-CCCA secure if the underlying IBE scheme is IND-CCA secure, the OTS is UF secure, and the secret sharing scheme is information-theoretically secure.*

Proof. (*Sketch*) This construction is very similar to the WF-KEM construction that is proven secure in Theorem 6.2. The only difference in the ciphertext is the inclusion of an additional component which corresponds to the PKE encapsulation of a random secret.

Furthermore, in the credential security model for an EFWF-KEM, the adversary knows all the parameters for the underlying PKE, including the secret key, so the additional component in the challenge ciphertext is adding no additional security. We need only to show that the public-key component also does not degrade the security of the construction. To do this we analyse how the adversary could obtain additional advantage, and show that this is not feasible.

Global Parameters: The global parameters for the PKE component are completely independent from the remaining global parameters, so the adversary can obtain no additional advantage through them.

Challenge Oracle: The challenge ciphertext is constructed using an information-theoretically secure 2-out-of-2 splitting of the secret key. One of the shares is handed over to the adversary, since it can open up the public-key component. But this provides no information whatsoever about the encapsulated secret, unless the adversary can learn something about the other share.

Decapsulation Oracle: The strategy used in Theorem 6.2 to handle decapsulation queries can also be used in this case.

Credential Extraction Oracle: Any credential extraction oracle queries that the adversary may perform will provide no more advantage than it would obtain against the WF-KEM construction, as the IBE component is completely independent of the PKE component.

To summarise, a complete proof of the IND-CCCA security of this EFWF-KEM construction would be almost identical to the proof in Theorem 6.2, the only difference being the fact that the B_1 and B_2 algorithms would need to include the PKE component in their interaction with the adversary. These algorithms would generate the additional parameters themselves, provide them to the adversary, and use them to answer any related decapsulation queries that the adversary might make. □

Again for computational secret sharing we have the following result.

Theorem 6.5. *The above EFWF-KEM construction is* (m,n)*-IND-CCCA*$^{-}$ *and* (m,n)*-IND-CCA secure if the underlying PKE is IND-CCA secure, the underlying IBE is IND-CCA secure, the OTS is UF secure, and the secret sharing scheme is IND-SSA secure. More precisely, for any PPT adversary A against the WF-KEM, there are PPT adversaries* B_1*,* B_2 *and* B_3 *against the OTS, the IBE/PKE scheme and the computational secret sharing scheme such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCCA}^{-}}_{\mathtt{EFWF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2mn^2 \cdot \mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{IBE}}(B_2) + \mathrm{Adv}^{\mathtt{IND-SSA}}_{\mathtt{SS}}(B_3),$$

$$\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{EFWF-KEM}}(A) \leq 2\mathrm{Adv}^{\mathtt{UF}}_{\mathtt{OTS}}(B_1) + 2\mathrm{Adv}^{\mathtt{IND-CCA}}_{\mathtt{PKE}}(B_2).$$

Proof. (*Sketch*) The only difference introduced by allowing for a computational secret sharing scheme resides on the credential security result, as recipient security is guaranteed by the OTS and PKE schemes. The same argument presented for the WF-KEM construction in Theorem 6.3 applies here. □

6.3 Discussion

The main contribution of our work in this chapter is that, through our generic constructions and composition theorems, and using underlying components which

achieve the required levels of security in the standard model, we are able to obtain the first WF-ENC and EFWF-ENC schemes which are provably secure in the standard model.

There are, however, other interesting aspects to the results presented in the previous sections, which we now discuss.

Relation with the original construction in [3]: The concrete escrow-free workflow scheme in [3] is defined as a full encryption scheme, although internally it is structured as a KEM-DEM construction. The basic building-block in the KEM part is a weak version of the IBE scheme by Boneh and Franklin [23]. Chosen ciphertext security is achieved globally through a transformation akin to that used in the KEM constructions in [38], which is valid in the random oracle model. Here, we require fully chosen ciphertext secure individual components, and the way we achieve global CCA security in the standard model comes from the IBE to PKE transformation in [29], adapted to multiple encryption in [43].

Our constructions do inherit the combination of a secret sharing scheme, an IBE scheme and a PKE scheme. However, if we allow for computational secret sharing, then we can only achieve IND-CCCA^{-} security. This is true even if the underlying secret sharing scheme tolerates adaptive chosen share attacks. This is the main difference between the security of our construction and that in [3]. Intuitively this can be explained as follows. Using random oracle heuristic one can perform a *late binding* between challenge share values and the challenge ciphertext. This makes it possible to construct the challenge without explicitly knowing the shares, and directly map the adversary's credential extraction queries to external calls to a share extraction oracle. The standard model does not allow the same proof strategy, so we cannot prove the security of our constructions against adaptive credential extracting attackers unless we adopt perfect secret sharing. This will

only be an issue in terms of the overall efficiency of the constructions, which we discuss below.

Finally, it is interesting to note the effective application of the randomness-reuse technique, as described in Chapter 5, in the original scheme to achieve impressive computational and ciphertext length savings.

Relation with multiple encryption: This work builds on the general results by Dodis and Katz [43] for chosen ciphertext security of multiple encryption. However our constructions require that we extend these results in three different aspects: (1) we consider adaptive user corruption attacks; (2) we investigate generalised access structures; and (3) we require a mix of identity-based and public-key encryption techniques. Our results imply that equivalent extensions can be derived in the context of generic multiple encryption.

Relation with certificateless encryption: We will not explore this connection in detail due to space constraints. However, we do note the similarity between the security models of a CLKEM (see Chapter 7) and the EFWF-KEM security models introduced here. This similarity implies that a simplified version of our construction considering only one credential and a single authority can be seen as a CLKEM which can be proved IND-CCA secure against (normal) Type-I and Type-II adversaries.

Efficiency considerations: We analyse the efficiency of our constructions by looking at their computational load and ciphertext lengths. A high-level analysis shows that the computational weight associated with encapsulation and decapsulation is that of sharing the secret key, encrypting the n shares using the IBE scheme, possibly encrypting another secret key with the PKE scheme, and generating a one-time signature. The corresponding ciphertexts include the public sharing in-

formation, n IBE ciphertexts, possibly one PKE ciphertext, the OTS verification key and a signature string.

An obvious way to optimise the end-result is to choose underlying components which are themselves efficient. For example, adopting the IBE scheme of Sakai and Kasahara [30] one obtains a solution which is computationally more efficient than the original construction in [3]. However, there are three techniques which can further improve the efficiency of our constructions.

The enhanced IBE to PKE transformation proposed in [24], which replaces the OTS component by a MAC and a weak form of commitment has also been adapted to achieve chosen ciphertext security in [43] for a weak form of multiple encryption. It turns out that this weak form of multiple encryption is sufficient to allow an extension to WF-KEMs similar to what we achieved with the OTS-based technique. We chose not to include these results in this work as they lead to more involved proofs and they are less intuitive.

The randomness-reuse technique can also be applied in this context. However, as stated in Section 5.6, there is currently no IBE scheme which is IND-CCA secure in the standard model, and which allows reuse of randomness across identities. If we settle security in the ROM, for the fully secure version of the Boneh–Franklin IBE scheme, we can obtain bandwidth and computational (point multiplication) savings by reusing the first component in all IBE ciphertexts. Further improvements may be attainable by reusing the same randomness in the PKE component as in [3].

Our constructions can be easily adapted to work with IBE and PKE schemes extended to take labels as additional parameters, and bind them non-malleably to ciphertexts. This adaptation reduces to using the OTS verification key as the label. Potential benefits of this would arise from labelled IBE or PKE schemes which achieve this functionality more efficiently than the direct non-malleable labelling

that we adopted in our constructions [86].

As a final note on efficiency, we look at the potential benefits of using a computational secret sharing scheme rather than a perfect secret sharing scheme. The main advantage in this is to obtain share sizes which are smaller than the shared secret, which is important when the secret is large. For example, the scheme in [59] uses a perfect secret sharing scheme as an underlying component to split an auxiliary secret key. This key is then used to encrypt the results of partitioning the (large) secret using an information dispersal algorithm. This provides share sizes which asymptotically approach the optimal $|S|/n$ by detaching the size of the (large) secret from the input to the perfect secret sharing scheme. In our case this is an invalid argument, as the secrets we share are themselves the secret keys which will be used to encrypt data.

Chapter 7

Certificateless Key Encapsulation Mechanisms

In this chapter we extend the concept of key encapsulation mechanisms to the certificateless setting (a CLKEM). As done for the identity-based settings in Chapter 3, we prove an appropriate CLKEM-DEM composition theorem and then give a generic construction of secure CLKEMs based on weakly secure IBE and PKE schemes. The security proofs in this chapter are substantially updated versions of those which appeared in a joint with Kamel Bentahar, John Malone-Lee and Nigel Smart [20].

7.1 The CLKEM-DEM Paradigm

7.1.1 The CLKEM Primitive

We adapt the KEM definition of Section 2.3.3 to the case of the certificateless systems as introduced in Section 2.3.6. A CLKEM is specified by six PPT algorithms as follows.

- $\mathtt{Setup}(1^\kappa)$. This is a global setup algorithm, which takes as input the secu-

rity parameter 1^κ and returns the Key Generation Centre's (KGC) secret key $\mathtt{Msk}$ and domain parameters I including a master public key $\mathtt{Mpk}$ and descriptions of key space $\mathcal{K}_{\mathtt{CLKEM}}(I)$, ciphertext space $\mathcal{C}_{\mathtt{CLKEM}}(I)$. This algorithm is executed by the KGC, which publishes I.

- $\mathtt{Extract\text{-}Partial\text{-}Private\text{-}Key}(\mathtt{ID}, \mathtt{Msk}, I)$. A probabilistic algorithm that takes as input $\mathtt{Msk}$, I and an identifier string $\mathtt{ID} \in \{0,1\}^*$ representing a user's identity, and returns a partial private key D. This algorithm is run by the KGC, after verifying the user's identity.

- $\mathtt{Generate\text{-}User\text{-}Keys}(\mathtt{ID}, I)$. A probabilistic algorithm which takes an identity and the domain parameters and outputs a secret value $\mathtt{SK}$ and a public key $\mathtt{PK}$. This algorithm is run by a user to obtain a public key and a secret value which can be used to construct a full private key. The public key is published without certification.

- $\mathtt{Set\text{-}Private\text{-}Key}(D, \mathtt{SK}, I)$. A deterministic algorithm which takes as input a partial private key D and a secret value $\mathtt{SK}$ and returns the full private key S. Again, this algorithm is run by a user to construct the full private decryption key.

- $\mathbb{E}_{\mathtt{CLKEM}}(\mathtt{PK}, \mathtt{ID}, I)$. This is the probabilistic encapsulation algorithm. On input of $\mathtt{PK}$, $\mathtt{ID}$ and I this outputs a pair $(\mathtt{k}, \mathtt{c})$, where $\mathtt{k} \in \mathcal{K}_{\mathtt{CLKEM}}(I)$ is a key and $\mathtt{c} \in \mathcal{C}_{\mathtt{CLKEM}}(I)$ is the encapsulation of that key, or an error symbol $\perp$.

- $\mathbb{D}_{\mathtt{CLKEM}}(\mathtt{c}, S)$. This is the deterministic decapsulation algorithm. On input of $\mathtt{c}$ and S this outputs a key $\mathtt{k}$ or a failure symbol $\perp$.

To define the security model for CLKEMs we simply adapt the security model of Al-Riyami and Paterson [5] into the KEM framework.

IND-atk-x^s

1. $(\mathtt{Msk}, \mathtt{Mpk}) \leftarrow \mathtt{Setup}(1^\kappa)$
2. $(\mathtt{ID}^*, \mathrm{st}) \leftarrow A_1^{O_1}(\mathtt{Mpk}, \mathtt{aux})$
3. $(\mathtt{k}_0, \mathtt{c}^*) \leftarrow \mathbb{E}_{\mathtt{CLKEM}}(\mathtt{PK}^*, \mathtt{ID}^*, \mathtt{Mpk})$
4. $\mathtt{k}_1 \leftarrow \mathcal{K}_{\mathtt{CLKEM}}(I)$
5. $b \leftarrow \{0,1\}$
6. $b' \leftarrow A_2^{O_2}(\mathtt{k}_b, \mathtt{c}^*, \mathrm{st})$

$$\mathrm{Adv}^{\mathtt{IND-CCA-x}^s}_{\mathtt{CLKEM}}(A) := |2\Pr[b' = b] - 1|.$$

Here, as for CLEs, x is either I or II and s is $-$, the empty string or $+$, reflecting the strength of the attack. When performing the encapsulation in line three above the challenger uses the *current* public key $\mathtt{PK}^*$ of the entity with identifier $\mathtt{ID}^*$. Furthermore, in a Type-II attack $\mathtt{aux} := \mathtt{Msk}$. The various oracle accesses allowed are identical to those defined in Section 2.3.6, where we simply replace the word "decryption" with "decapsulation".

7.1.2 Combining CLKEMs with DEMs

Similarly to the identity-based composition theorem in Section 3.1.2, we prove our CLKEM-DEM composition theorem which allows one to construct IND-CCA secure certificateless encryption schemes from IND-CCA secure CLKEMs and FG-CCA secure DEMs. Again we assume that the key space output by the CLKEM corresponds to the key space required by the DEM, and that for all keys, all encapsulations decapsulate properly.

Theorem 7.1. *Let A be a PPT adversary against the hybrid certificateless scheme in the sense of IND-CCA-I and IND-CCA-II adversaries, then there exist PPT ad-*

versaries B_1 and B_2, whose running time is essentially that of A, such that

$$\begin{aligned}\mathrm{Adv}^{\mathtt{IND-CCA-I}}_{\mathtt{CLE}}(A) &\leq 2\mathrm{Adv}^{\mathtt{IND-CCA-I}}_{\mathtt{CLKEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2),\\ \mathrm{Adv}^{\mathtt{IND-CCA-II}}_{\mathtt{CLE}}(A) &\leq 2\mathrm{Adv}^{\mathtt{IND-CCA-II}}_{\mathtt{CLKEM}}(B_1) + \mathrm{Adv}^{\mathtt{FG-CCA}}_{\mathtt{DEM}}(B_2).\end{aligned}$$

The proof of this theorem is similar to that of Theorem 4.1. There is, however, an important difference when adapting Lemma 4.2 to the certificateless setting in strong models. We can no longer provide decryptions on replaced public keys and we shall require the adversary to provide the associated secret values, as defined in the normal Type-I model. This obstacle to proving natural composition theorems for strong models could be viewed as another argument against their relevance in practice. Also it can be easily deduced that analogous results hold for weak models.

We note that since an IND-CCA-I$^+$/II$^+$ secure CLKEM is clearly IND-CCA-I/II secure, the above result allows us to combine an IND-CCA-I$^+$/II$^+$ secure CLKEM with a secure DEM, so as to obtain an IND-CCA-I/II secure CLE scheme.

7.2 CLKEM Construction

Our generic construction of a CLKEM can now be given as follows.

- Let $(\mathbb{G}_{\mathtt{PKE}}, \mathbb{E}_{\mathtt{PKE}}, \mathbb{D}_{\mathtt{PKE}})$ be a public-key encryption scheme.
- Let $(\mathbb{G}_{\mathtt{IBE}}, \mathbb{X}_{\mathtt{IBE}}, \mathbb{E}_{\mathtt{IBE}}, \mathbb{D}_{\mathtt{IBE}})$ be an identity-based encryption scheme.

We define our six algorithms as follows. The algorithm `Setup` is defined to be equal to $\mathbb{G}_{\mathtt{IBE}}$ and I is set to be `Mpk`. The algorithm partial private key extraction returns D, the output from $\mathbb{X}_{\mathtt{IBE}}(\mathtt{Msk}, \mathtt{ID})$. The values returned by the user key generation algorithm are simply the outputs `SK` and `PK` from $\mathbb{G}_{\mathtt{PKE}}$. The algorithm Set-Private-Key returns the pair $S = (D, \mathtt{SK})$. Finally, using a hash function H, encapsulation and decapsulation work as follows.

$\mathbb{E}_{\texttt{CLKEM}}(\texttt{PK}, \texttt{ID}, \texttt{Mpk})$

- $\texttt{m}_1 \leftarrow \mathcal{M}_{\texttt{PKE}}(\texttt{PK})$
- $\texttt{m}_2 \leftarrow \mathcal{M}_{\texttt{IBE}}(I)$
- $\texttt{c}_1 \leftarrow \mathbb{E}_{\texttt{PKE}}(\texttt{m}_1, \texttt{PK})$
- $\texttt{c}_2 \leftarrow \mathbb{E}_{\texttt{IBE}}(\texttt{m}_2, \texttt{ID}, \texttt{Mpk})$
- $\texttt{k} \leftarrow H(\texttt{c}_1, \texttt{PK}, \texttt{m}_1, \texttt{m}_2)$
- $\texttt{c} \leftarrow (\texttt{c}_1, \texttt{c}_2)$
- Return $\texttt{c}$

$\mathbb{D}_{\texttt{CLKEM}}(\texttt{c}, S)$

- $(\texttt{c}_1, \texttt{c}_2) \leftarrow \texttt{c}$
- $(D, \texttt{SK}) \leftarrow S$
- $\texttt{m}_1 \leftarrow \mathbb{D}_{\texttt{PKE}}(\texttt{c}_1, \texttt{SK})$
- $\texttt{m}_2 \leftarrow \mathbb{D}_{\texttt{IBE}}(\texttt{c}_2, D)$
- If $\texttt{m}_1 = \perp$ or $\texttt{m}_2 = \perp$ return $\perp$
- $\texttt{k} \leftarrow H(\texttt{c}_1, \texttt{PK}, \texttt{m}_1, \texttt{m}_2)$
- Return $\texttt{k}$

7.2.1 Security Against Type-I$^+$ Adversaries

In this section we shall prove that Type-I$^+$ security of our generic CLKEM construction rests both on the security of the identity-based component of the scheme and on the security of the public-key component. Throughout the proof we shall be dealing with the OW-cCPA^{++} security model. We refer the reader to Section 2.3.2 for the details.

Theorem 7.2. *For any Type-I$^+$ adversary A against our generic CLKEM construction in the random oracle model, there is an adversary B_1 against the identity-based encryption scheme in the OW-CCA sense and an adversary B_2 against the public-key scheme in the OW-CPA^{++} sense such that:*

$$\text{Adv}^{\texttt{IND-CCA-I}^+}_{\texttt{CLKEM}}(A) \leq 2(q_H + q_D)\text{Adv}^{\texttt{OW-CCA}}_{\texttt{IBE}}(B_1) + 2(q_{PK} + q_D + 1)\text{Adv}^{\texttt{OW-cCPA}^{++}}_{\texttt{PKE}}(B_2).$$

Here q_H, q_{PK}, and q_D denote the maximum number of calls that A places to the random oracle H, public key request, and decapsulation oracles respectively.

Proof. Let A denote a Type-I$^+$ adversary against our CLKEM as specified in the

statement of the theorem. We let $\mathtt{ID}^*$ denote the challenge identity chosen by A after its first stage. We shall denote the target encapsulation by $\mathtt{c}^* = (\mathtt{c}_1^*, \mathtt{c}_2^*)$ which encapsulates the key $\mathtt{k}_1^*$. Let $\mathtt{m}_1^*$ denote the message encrypted in $\mathtt{c}_1^*$ under $\mathtt{PK}^*$, the public key of $\mathtt{ID}^*$ at the time the challenge ciphertext is created, and let $\mathtt{m}_2^*$ denote the message encrypted in $\mathtt{c}_2^*$ under $\mathtt{ID}^*$. Let $\mathtt{k}_0^*$ denote a random key selected from the co-domain of H, and let b denote a random bit; both outside the view of the adversary. In the second stage of the game we let $\mathtt{k}^* = \mathtt{k}_b^*$ denote the key given to the adversary and we let b' denote the bit returned by the adversary.

Security is proved using two games $\mathtt{Game}_0$ and $\mathtt{Game}_1$. In each game $\mathtt{Game}_i$ we let S_i denote the event that $b' = b$.

We let $\mathtt{Game}_0$ denote the original attack game and so by definition

$$(1/2) \cdot \mathrm{Adv}^{\mathtt{IND-CCA-I}^+}_{\mathtt{CLKEM}}(A) = |\Pr[S_0] - 1/2|.$$

In $\mathtt{Game}_1$ we replace the public key request, public key replacement, full private key extraction, the hash function oracle, and the decapsulation oracles by the following.

- **Public Key Request:** We keep a list L_{PK} of triples $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ which is at most of length $q_{PK} + q_{SK} + q_X + q_R + q_D + 1$, where q_R, q_{SK} and q_X are the number queries placed to the public key request, full private key and partial private key extraction oracles. On input an identity $\mathtt{ID}$, we check whether this identity already appears on the list, if so we respond with either $\mathtt{PK}$. Otherwise we call $\mathbb{G}_{\mathtt{PKE}}$ to obtain a new pair $(\mathtt{SK}, \mathtt{PK})$ insert $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ onto the list and then return $\mathtt{PK}$.

- **Public Key Replacement:** If A wishes to replace the public key for user $\mathtt{ID}$ with $\mathtt{PK}'$ then we search the L_{PK} list for an entry corresponding to $\mathtt{ID}$ and replace this entry with $(\mathtt{ID}, \mathtt{PK}', \perp)$. If no such entry exists then we add

$(\mathtt{ID}, \mathtt{PK}', \perp)$ to the list L_{PK}.

- **Full Private Key Extraction:** The challenger answers these queries by obtaining the appropriate secret value from the L_{PK} list and concatenating it with the answer obtained by querying the partial private key extraction oracle.

- **Hash Function:** We keep a list L_H of tuples $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ of length at most $q_H + q_D$. If this oracle is called with input $(\mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ we perform the following steps:

  - If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ is on L_H, we respond with $\mathtt{k}$.
  - If there is some $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ on L_H such that $\mathtt{c}_1$ is the encryption of $\mathtt{m}_1$ under the key $\mathtt{PK}$ we respond with $\mathtt{k}$ (and update this entry to make it $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$). Note, this step requires the complete plaintext/ciphertext checking oracle.
  - Otherwise we generate $\mathtt{k}$ at random from the co-domain of H, we place $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ onto L_H and respond with $\mathtt{k}$.

- **Decapsulation Queries:** On input of $(\mathtt{c}_1, \mathtt{c}_2)$ and $\mathtt{ID}$, the simulator for algorithm A can decrypt the component $\mathtt{c}_2$ to obtain $\mathtt{m}_2$, since it knows the master key for the identity-based scheme. It returns $\perp$ if $\mathtt{m}_2 = \perp$. Then, by making a call to the public key request oracle we can obtain the public/private key pair $(\mathtt{PK}, \mathtt{SK})$ from the list L_{PK} corresponding to the identity $\mathtt{ID}$. There are two cases:

  - $\mathtt{SK} \neq \perp$, in which case the public key has not been replaced. We can then use $\mathtt{SK}$ to decrypt $\mathtt{c}_1$ to obtain $\mathtt{m}_1$, returning $\perp$ if $\mathtt{m}_1 = \perp$. The simulator for hash function H can then be called on the input $(\mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ so as to obtain the encapsulated key $\mathtt{k}$.

– $\mathtt{SK} = \perp$, in which case the public key has been replaced. We call the ciphertext validity oracle on $(\mathtt{c}, \mathtt{PK})$ and return $\perp$ if $\mathtt{c}_1$ is invalid. We then search L_H to find an entry $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ such that $\mathtt{c}_1$ is the encryption of $\mathtt{m}_1$ under the key $\mathtt{PK}$. Note again that this uses the complete plaintext/ciphertext checking oracle. If such an entry exists then we return $\mathtt{k}$. Otherwise, if $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ appears on L_H we return $\mathtt{k}$, else we generate $\mathtt{k}$ at random from the co-domain of H, place $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ onto L_H and return $\mathtt{k}$.

Since A is working in the random oracle model its view in two games are identical:

$$\Pr[S_0] = \Pr[S_1].$$

Before proceeding we define three events.

- R: The event that A replaces the public key for $\mathtt{ID}^*$ before the challenge ciphertext is issued.
- E: The event that A extracts the partial private key for $\mathtt{ID}^*$.
- $\mathtt{Ask}$: The event that the simulator for H is called with input $(\mathtt{c}_1^*, \mathtt{PK}^*, \mathtt{m}_1^*, \mathtt{m}_2^*)$.

We immediately have the following.

$$\begin{aligned} \Pr[S_1] &= \Pr[S_1 \wedge R] + \Pr[S_1 \wedge \neg R] \\ &= \Pr[S_1|R]\Pr[R] + \Pr[S_1|\neg R](1 - \Pr[R]). \end{aligned}$$

Also,

$$\Pr[S_1|R] = \Pr[S_1 \wedge \neg E|R].$$

The last equality above follows from the fact that, by definition of a Type-I$^+$ adversary, if R occurs then E is forbidden.

Now,

$$\begin{aligned}\Pr[S_1 \wedge \neg E|R] &= \Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}|R] + \Pr[S_1 \wedge \neg E \wedge \neg \mathtt{Ask}|R] \\ &= \Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}|R] + \frac{1}{2}.\end{aligned}$$

The final equality follows from the fact that, if the query $(\mathtt{c}_1^*, \mathtt{PK}^*, \mathtt{m}_1^*, \mathtt{m}_2^*)$ is never made to the simulator for H, then A can have no advantage.

We also have

$$\begin{aligned}\Pr[S_1|\neg R] &= \Pr[S_1 \wedge \mathtt{Ask}|\neg R] + \Pr[S_1 \wedge \neg \mathtt{Ask}|\neg R] \\ &= \Pr[S_1 \wedge \mathtt{Ask}|\neg R] + \frac{1}{2}.\end{aligned}$$

Again, the last equality follows from the fact that, if the query $(\mathtt{c}_1^*, \mathtt{PK}^*, \mathtt{m}_1^*, \mathtt{m}_2^*)$ is never made to the simulator for H, then A can have no advantage.

Putting the above equalities together we have:

$$\begin{aligned}|2\Pr[S_1] - 1| &= |2\Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}|R]\Pr[R] + \Pr[R] + \\ &\quad +2\Pr[S_1 \wedge \mathtt{Ask}|\neg R]\Pr[\neg R] + 1 - \Pr[R] - 1| \\ &= |2\Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}|R]\Pr[R] + 2\Pr[S_1 \wedge \mathtt{Ask}|\neg R]\Pr[\neg R]| \\ &\leq 2\Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}|R]\Pr[R] + 2\Pr[S_1 \wedge \mathtt{Ask}|\neg R]\Pr[\neg R] \\ &\leq 2\Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}] + 2\Pr[S_1 \wedge \mathtt{Ask}|\neg R].\end{aligned}$$

We now describe an algorithm B_1 to break the assumed OW-CCA security of the identity-based encryption scheme used in the construction when the event $S_1 \wedge \neg E \wedge \mathtt{Ask}$ occurs.

This algorithm runs A in a similar manner to how A is run in $\mathtt{Game}_1$. The first differences is how we respond to A's partial private key extraction and decapsulation queries in the construction of B_1.

- **Partial Private Key Extraction:** The challenger answers these queries using the IBE extraction oracle provided in the OW-CCA game. Note that the simulation will not fail as the event E will not occur.

- **Decapsulation Queries:** We introduce an additional list L'_H. This list is initially empty, it is updated by the new decapsulation oracle as described below.

  Suppose that we are responding to a query $(\mathtt{ID}, (\mathtt{c}_1, \mathtt{c}_2))$. If $\mathtt{ID} \neq \mathtt{ID}^*$ or $\mathtt{c}_2 \neq \mathtt{c}_2^*$ we respond as in $\mathtt{Game}_1$ except that, rather than using knowledge of the master key for the identity-based scheme which we no longer have, we decrypt $\mathtt{c}_2$ using the decryption oracle provided to B_1. Otherwise, we make a call to the public key request oracle to obtain $(\mathtt{PK}, \mathtt{SK})$ from the list L_{PK} corresponding to the identity $\mathtt{ID}$. There are two cases.

  - If $\mathtt{SK} \neq \perp$, we decrypt $\mathtt{c}_1$, returning $\perp$ on failure. If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1)$ appears on L'_H we return $\mathtt{k}$. Otherwise we generate $\mathtt{k}$ at random from the co-domain of H, add $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1)$ to L'_H and respond with $\mathtt{k}$. Note that the simulation is inconsistent if $(\mathtt{k}', \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2^*)$ with $\mathtt{k}' \neq \mathtt{k}$ is (going to be) on L_H.
  - If $\mathtt{SK} = \perp$, we first check the validity of $\mathtt{c}_1$ using the complete ciphertext validity checking oracle. If this oracle returns 0, we return $\perp$. Now if there is a $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1)$ in L'_H such that the complete plaintext/ciphertext checking oracle returns 1 on $(\mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1)$ we return $\mathtt{k}$. Else we check if $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp)$ appears on L'_H, and if so return $\mathtt{k}$. Otherwise we generate $\mathtt{k}$ at random from the co-domain of H, add the tuple $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp)$ to L'_H and respond with $\mathtt{k}$. Note that the simulation is inconsistent if $(\mathtt{k}', \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2^*)$ with $\mathtt{k}' \neq \mathtt{k}$ is (going to be) on L_H.

To generate the challenge citphertext for A we proceed as usual to compute $\mathtt{c}_1^*$,

we obtain $\mathtt{c}_2^*$ by relaying $\mathtt{ID}^*$ output by A to B_1's challenge oracle and we choose $\mathtt{k}^*$ at random from the co-domain of H.

Finally, at the end of A's execution, we choose a random input from L_H and output the last component, $\mathtt{m}_2$, as B_1's attempt to recover the plaintext within $\mathtt{c}_2^*$. Let us define the event

- $\mathtt{Ask}'$: The event that an entry of the form $(\mathtt{k}, \mathtt{c}, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2^*)$ or $(\mathtt{k}, \mathtt{c}, \mathtt{PK}, \perp, \mathtt{m}_2^*)$ appears on L_H.

Now, as long $\mathtt{Ask}'$ does not occur, A is run by B_1 in exactly the same manner as A is run in $\mathtt{Game}_1$; moreover, if the event $\mathtt{Ask}'$ occurs B_1 succeeds to recover the plaintext within $\mathtt{c}_2^*$ with probability at least $1/(q_H + q_D)$ since there are at most $(q_H + q_D)$ entries in L_H. However, $\mathtt{Ask}'$ is implied by $\mathtt{Ask}$. This tells us that

$$\Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}] \leq \Pr[S_1 \wedge \neg E \wedge \mathtt{Ask}'] \leq (q_H + q_D)\mathrm{Adv}_{\mathtt{IBE}}^{\mathtt{OW-CCA}}(B_1).$$

We complete the proof by describing an adversary B_2 of the public-key scheme used in our construction when the event $S_1 \wedge \mathtt{Ask}$ occurs and the event $\neg R$ is assumed to hold throughout.

The adversary B_2 is given a public key $\mathtt{PK}^*$ for which it wishes to recover a message from a ciphertext. To construct B_2 we run A in similar manner to how A is run in $\mathtt{Game}_1$. The oracles that are modified are described below.

- **Public Key Request:** At the very beginning of the simulation we choose i uniformly at random from $\{1, \ldots, q_{PK} + q_D + 1\}$. We maintain a list L_{PK} of tuples $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ of length at most $q_{PK} + q_R + q_{SK} + q_D + 1$. We have two cases when responding to a query $\mathtt{ID}$.
  - If we are responding to the i-th public key request, made either by A directly or by the decapsulation oracle, or made by the challenge

encryption oracle, we respond with $\mathtt{PK}^*$ and add $(\mathtt{ID}, \mathtt{PK}^*, \perp)$ to L_{PK}. We do not need to monitor queries from public key replacement and full private key extraction oracles as these are not permitted on $\mathtt{ID}^*$ in stage 1 (we have conditioned on $\neg R$).

– Otherwise, we check whether this identity already appears on the list, if so we respond with either PK or SK as appropriate and, if not, we call $\mathbb{G}_{\mathtt{PKE}}$ to obtain a new pair $(\mathtt{SK}, \mathtt{PK})$ insert $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ onto the list and then return PK.

- **Hash Function:** We modify how the hash function operates after the challenge ciphertext has been issued. Suppose that we are responding to a query $(\mathtt{c}_1^*, \mathtt{PK}^*, \mathtt{m}_1, \mathtt{m}_2)$, where $\mathtt{c}_1^*$ is the first component of the challenge encapsulation. Before proceeding as in $\mathtt{Game}_1$ we check if the complete plaintext/ciphertext checking oracle returns 1 on input $(\mathtt{c}_1^*, \mathtt{PK}^*, \mathtt{m}_1)$. If so we output $\mathtt{m}_1$ and terminate the simulation.

To generate the challenge ciphertext for A first call the public key request oracle. If we do not receive $\mathtt{PK}^*$ in response we abort the simulation. If we do receive $\mathtt{PK}^*$ we proceed as usual to compute $\mathtt{c}_2^*$, we obtain $\mathtt{c}_1^*$ by calling B_2's challenge encryption and $\mathtt{k}^*$ at random from the co-domain of H.

Now, conditioned on the event $\neg R$, when we are generating the challenge ciphertext we obtain $\mathtt{PK}^*$ from the public key request oracle with probability at least $1/(q_{PK} + q_D + 1)$. Assuming this is so, A is run by B_2 in exactly the same way that A is run in $\mathtt{Game}_1$ up until the event Ask occurs and, moreover, if the event Ask occurs, B_2 succeeds due to the simulation of H above. We conclude that

$$\Pr[S_1 \wedge \mathtt{Ask} | \neg R] \leq (q_{PK} + q_D + 1)\mathrm{Adv}_{\mathtt{PKE}}^{\mathtt{OW-cCPA}^{++}}(B_2).$$

□

7.2.2 Security Against Type-II$^+$ Adversaries

In this section we shall prove that Type-II$^+$ security of our generic CLKEM construction rests solely on the security of the public-key component of the scheme.

Theorem 7.3. *For any Type-II$^+$ adversary A against our generic CLKEM construction in the random oracle model, there is an adversary B_2 against the public-key scheme in the OW-CPA^{++} sense such that:*

$$\mathrm{Adv}^{\mathtt{IND-CCA-II}^+}_{\mathtt{CLKEM}}(A) \leq 2q_T \mathrm{Adv}^{\mathtt{OW-cCPA}^{++}}_{\mathtt{PKE}}(B_2).$$

Here $q_T = q_{PK} + q_D + 1$ and various q_*'s are as in Theorem 7.2.

Proof. Let A denote a Type-II$^+$ adversary against our CLKEM as specified in the statement of the theorem. Security is proved via games $\mathtt{Game}_0$, $\mathtt{Game}_1$ and $\mathtt{Game}_2$. We define $\mathtt{ID}^*$, $\mathtt{PK}^*$, $\mathtt{c}^* = (\mathtt{c}^*_1, \mathtt{c}^*_2)$, $\mathtt{m}^*_1$, $\mathtt{m}^*_2$, $\mathtt{k}^*_0$, $\mathtt{k}^*_1$, b, b', S_i and $\mathtt{Ask}$ exactly as in the proof of Theorem 7.2.

We let $\mathtt{Game}_0$ denote the original attack game and so

$$(1/2) \cdot \mathrm{Adv}^{\mathtt{IND-CCA-II}^+}_{\mathtt{CLKEM}}(A) = |\Pr[S_0] - 1/2|.$$

In $\mathtt{Game}_1$ we replace the public key request, public key replacement, full private key extraction and the hash function oracles by the following.

- **Public Key Request:** We keep a list L_{PK} of triples $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ which is at most of length $q_{PK} + q_R + q_{SK} + q_D + 1$. On input an identity $\mathtt{ID}$ we check whether this identity already appears on the list, if so we respond with $\mathtt{PK}$. Otherwise we call $\mathbb{G}_{\mathtt{PKE}}$ to obtain a new pair $(\mathtt{SK}, \mathtt{PK})$ insert $(\mathtt{ID}, \mathtt{PK}, \mathtt{SK})$ onto the list and then return $\mathtt{PK}$.

- **Public Key Replacement:** If A wishes to replace the public key for user $\mathtt{ID}$ with $\mathtt{PK}'$ then we search the L_{PK} list for an entry corresponding to $\mathtt{ID}$

and replace this entry with $(\mathtt{ID}, \mathtt{PK}', \perp)$. If no such entry exists then we add $(\mathtt{ID}, \mathtt{PK}', \perp)$ to the list L_{PK}.

- **Full Private Key Extraction:** The challenger answers these queries by obtaining the appropriate secret value from the L_{PK} list and concatenating it with the partial private key.

- **Hash Function:** We keep a list L_H of tuples $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ of length at most $q_H + q_D$. If this oracle is called with input $(\mathtt{c}, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ we see whether this pair already appears on the list, if so we respond with the appropriate value of $\mathtt{k}$. Otherwise we generate $\mathtt{k}$ at random from the co-domain of H, we place $(\mathtt{k}, \mathtt{c}, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ onto the list and return $\mathtt{k}$.

Since A is working in the random oracle model, the two games are identical. We have

$$\begin{aligned} \Pr[S_0] &= \Pr[S_1] \\ &= \Pr[S_1 \wedge \mathtt{Ask}] + \Pr[S_1 \wedge \neg \mathtt{Ask}] \\ &= \Pr[S_1 \wedge \mathtt{Ask}] + \frac{1}{2}. \end{aligned}$$

This last equality holds since H is a random oracle; if A does not make the critical query then it is able to determine whether or not $\mathtt{k}_b^*$ is the encapsulated key with probability at most $1/2$. Hence we have

$$|\Pr[S_0] - 1/2| = \Pr[S_1 \wedge \mathtt{Ask}] \leq \Pr[S_1 | \mathtt{Ask}].$$

In $\mathtt{Game}_2$ a random value i is chosen from $\{1, \ldots, q_T\}$, where $q_T = q_{PK} + q_D + 1$. In this game, we abort if the i-th query made either by A directly or by the decapsulation oracle or by the challenge oracle is not on the identity $\mathtt{ID}^*$[1].

[1] Again, we do not need to monitor queries from public key replacement and full private key

Let F_1 denote the abort event described above when considered in $\mathtt{Game}_1$. Then clearly $\Pr[\neg F_1] \geq 1/q_T$. In addition we have $\Pr[S_1|\mathtt{Ask} \wedge \neg F_1] = \Pr[S_2|\mathtt{Ask}]$. This gives us

$$\Pr[S_2|\mathtt{Ask}] = \Pr[S_1|\mathtt{Ask}] \cdot \Pr[\neg F_1] \geq \frac{\Pr[S_1|\mathtt{Ask}]}{q_T}.$$

The first equality above follows from the observations that $\Pr[S_1 \wedge \mathtt{Ask} \wedge \neg F_1] = \Pr[S_1 \wedge \mathtt{Ask}] \cdot \Pr[\neg F_1]$ and $\Pr[\mathtt{Ask} \wedge \neg F_1] = \Pr[\mathtt{Ask}]$.

We claim that

$$\Pr[S_2|\mathtt{Ask}] = \mathrm{Adv}^{\mathtt{OW-cCPA}^{++}}_{\mathtt{PKE}}(B_2)$$

for an algorithm B_2.

Algorithm B_2 runs as follows. It generates $(\mathtt{Msk}, \mathtt{Mpk})$ by running $\mathbb{G}_{\mathtt{IBE}}(1^{\kappa})$. When A outputs $\mathtt{ID}^*$, algorithm B_2 calls its challenge oracle to obtain $\mathtt{c}_1^*$. It generates $\mathtt{m}_2 \leftarrow \mathcal{M}_{\mathtt{IBE}}(I)$ and sets $\mathtt{c}_2^* \leftarrow \mathbb{E}_{\mathtt{IBE}}(\mathtt{m}_2, \mathtt{ID}^*, \mathtt{Mpk}; r)$. Algorithm B_2 returns $(\mathtt{k}^*, \mathtt{c}_1^*, \mathtt{c}_2^*)$ for a random $\mathtt{k}^*$ from the co-domain of H as the challenge. Except the following alterations, algorithm B_2 answers the oracle calls of the algorithm A just as the oracles do in $\mathtt{Game}_2$.

- **Public Key Request:** The i-th query made either by A directly or by the decapsulation oracle or by the challenge oracle is replaced by $(\mathtt{ID}^*, \mathtt{PK}^*, \perp)$, where $\mathtt{ID}^*$ is the challenge identity output by A_1 and $\mathtt{PK}^*$ is the input public key for algorithm B_2. Note that, by the restrictions in the Type-II$^+$ model, we will not need to extract the full private key for $\mathtt{ID}^*$. Also $\mathtt{PK}^*$ cannot be replaced in the first stage.

- **Decapsulation Queries:** On input of $(\mathtt{c}_1, \mathtt{c}_2)$ and $\mathtt{ID}$, the simulator for algorithm A can decrypt the component $\mathtt{c}_2$ to obtain $\mathtt{m}_2$, since it knows the master key for the identity-based scheme. If $\mathtt{m}_2 = \perp$, it returns $\perp$. Then, by making a call to the public key request oracle we can obtain the public/private key

extraction oracles as these are not permitted on $\mathtt{ID}^*$ in stage 1.

pair $(\mathtt{PK}, \mathtt{SK})$ from the list L_{PK} corresponding to the identity $\mathtt{ID}$. If $\mathtt{SK} \neq \perp$, it can then decrypt $\mathtt{c}_1$ to obtain $\mathtt{m}_1$, returning $\perp$ if $\mathtt{m}_1 = \perp$. The hash function H can then be called on the input $(\mathtt{c}, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ so as to obtain the encapsulated key $\mathtt{k}$, modifying the list L_H as above.

If $\mathtt{SK} = \perp$, we perform the following steps:

- If $\mathtt{c}_1$ is not a valid ciphertext, which can be determined via the complete ciphertext validity oracle, we return $\perp$.
- If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ is on the list L_H then we check whether $\mathtt{c}_1$ is a valid encryption of $\mathtt{m}_1$, using the complete plaintext/ciphertext checking oracle. If so we output $\mathtt{k}$.
- If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ appears on L_H, we return $\mathtt{k}$, else we pick $\mathtt{k}$ at random, place $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ onto the L_H and return $\mathtt{k}$.

- **Hash Function:** Here we modify the oracle to make it compatible with the above decapsulation oracle. If H is called with input $(\mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ then we respond as follows:
  - If $(\mathtt{c}_1, \mathtt{PK}) = (\mathtt{c}_1^*, \mathtt{PK}^*)$ and the complete plaintext/ciphertext checking oracle returns 1 on $(\mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1)$, we return $\mathtt{m}_1$ and terminate the simulation.
  - If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ is on the list L_H then we output $\mathtt{k}$.
  - If $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \perp, \mathtt{m}_2)$ is on the list L_H and $\mathtt{c}_1$ is a valid encryption of $\mathtt{m}_1$ then we output $\mathtt{k}$ (and update the entry to read $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$). This uses the complete plaintext/ciphertext checking oracle.
  - Else we pick $\mathtt{k}$ at random, place $(\mathtt{k}, \mathtt{c}_1, \mathtt{PK}, \mathtt{m}_1, \mathtt{m}_2)$ onto L_H and return $\mathtt{k}$.

With these simulations algorithm A cannot notice the difference between running in $\mathtt{Game}_2$ and running as a subroutine for algorithm B_2. Since we have conditioned on the event $\mathtt{Ask}$, algorithm B_2 will recover $\mathtt{m}_1^*$ successfully due to step 1 of the simulation of the hash function H above. Therefore:

$$\begin{aligned}
\mathrm{Adv}_{\mathrm{PKE}}^{\mathtt{OW-cPA}^{++}}(B_2) &= \Pr[S_2|\mathtt{Ask}] \\
&\geq \frac{\Pr[S_1|\mathtt{Ask}]}{q_T} \\
&\geq \frac{|\Pr[S_0] - 1/2|}{q_T} \\
&= \frac{\mathrm{Adv}_{\mathrm{CLKEM}}^{\mathtt{IND-CCA-II}^{+}}(A)}{2q_T}.
\end{aligned}$$

□

Following our work, Libert and Quisquater [63] presented a certificateless analogue of the Fujisaki–Okamoto transformation. Given a CLE scheme with IND-CPA-I and IND-CPA-II security, they construct a new CLE scheme which is secure in the IND-CCA-I^{+} and IND-CCA-II$^{=}$ sense. This new transformation, similarly to our construction, hashes the public key and identity of the receiver when computing the random coins of the encryption algorithm. The authors also present a generic construction of an IND-CPA-x secure CLE scheme based on an IND-CPA secure IBE and an IND-CPA secure PKE scheme.

Strongly secure CLE schemes in the standard model were recently presented by Dent et al. in [41]. Two constructions are given in their paper. The first construction, which utilises the machinery of zero-knowledge proofs akin to the construction in [65], although quite general, is not practical. On the other hand, the second construction, which is based on the IBE scheme of Waters [88], is efficient.

7.3 Further Problems

The work in this chapter could be extended in a number of ways:

1. Give a CLKEM based on Dent's construction in Section 2.4.3, together with a (non-modular) proof that the resulting CLKEM-DEM construction achieves strong security if the DEM is secure. This construction could also eliminate the need for various oracles in the cCPA^{++} model.

2. More generally, extend the CLKEM-DEM framework to the strong security models.

3. Optimise (generic) constructions of CLE schemes which use a PKE and an IBE as building-blocks via randomness-reuse.

Chapter 8

Certificateless Signcryption

We saw in Section 2.3.6 that certificateless cryptography inherits from identity-based techniques a solution to the certificate management problem in public-key encryption, whilst removing the secret key escrow functionality inherent to the identity-based setting. Signcryption schemes [91] achieve confidentiality and authentication simultaneously by combining public-key encryption and digital signatures, offering better overall performance and security. In this chapter, we introduce the notion of certificateless signcryption and present an efficient construction which guarantees security under insider attacks, and therefore provides forward secrecy and non-repudiation. The scheme is shown to be secure in the random oracle model under a variant of the computational bilinear Diffie–Hellman assumption. The work in this chapter first appeared in a joint paper with Manuel Barbosa in [8].

8.1 Definition and Security Models

A certificateless signcryption (CLSC) scheme is defined by a six-tuple of PPT algorithms. Four of these algorithms, the ones corresponding to key management operations, are identical to those defined for certificateless encryption as described

in Section 2.3.6:

1. $\texttt{Setup}(1^\kappa)$. This is a global setup algorithm, which takes as input the security parameter 1^κ and returns the Key Generation Centre's (KGC) secret key $\texttt{Msk}$ and domain parameters I including a master public key $\texttt{Mpk}$ and descriptions of message space $\mathcal{M}_{\texttt{CLSC}}(I)$, ciphertext space $\mathcal{C}_{\texttt{CLSC}}(I)$ and randomness space $\mathcal{R}_{\texttt{CLSC}}(I)$. This algorithm is executed by the KGC, which publishes I.

2. $\texttt{Extract-Partial-Private-Key}(\texttt{ID}, \texttt{Msk}, I)$. A probabilistic algorithm that takes as input $\texttt{Msk}$, I and an identifier string $\texttt{ID} \in \{0,1\}^*$ representing a user's identity, and returns a partial private key D. This algorithm is run by the KGC, after verifying the user's identity.

3. $\texttt{Generate-User-Keys}(\texttt{ID}, I)$. A probabilistic algorithm which takes an identity and the domain parameters and outputs a secret value x and a public key $\texttt{PK}$. This algorithm is run by a user to obtain a public key and a secret value which can be used to construct a full private key. The public key is published without certification.

4. $\texttt{Set-Private-Key}(D, x, I)$. A deterministic algorithm which takes as input a partial private key D and a secret value x and returns the full private key S. Again, this algorithm is run by a user to construct the full private decryption key.

The signcryption and de-signcryption algorithms are as follows:

5. $\texttt{Sc}(\texttt{m}, S_S, \texttt{ID}_S, \texttt{PK}_S, \texttt{ID}_R, \texttt{PK}_R, I; r)$. This is the probabilistic signcryption algorithm. On input a message $\texttt{m} \in \mathcal{M}_{\texttt{CLSC}}(I)$, sender's full private key S_S, identity $\texttt{ID}_S$ and public key $\texttt{PK}_S$, the receiver's identity $\texttt{ID}_R$ and public key $\texttt{PK}_R$, the global parameters I and possibly some randomness $r \in \mathcal{R}_{\texttt{CLSC}}(I)$, this algorithm outputs a ciphertext $\texttt{c} \in \mathcal{C}_{\texttt{CLSC}}(I)$ or an error symbol $\perp$.

6. $\mathtt{Dsc}(\mathtt{c}, S_R, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{ID}_S, \mathtt{PK}_S, I)$. This is the deterministic de-signcryption algorithm. On input a ciphertext $\mathtt{c}$, receiver's full private key S_R, identity $\mathtt{ID}_R$ and public key $\mathtt{PK}_R$, the sender's identity $\mathtt{ID}_S$ and public key $\mathtt{PK}_S$ and the global parameters I, this algorithm outputs a plaintext $\mathtt{m}$ or a failure symbol $\perp$.

The correctness of a CLSC scheme is defined in the usual way.

Note that, as for identity-based signcryption, and unlike standard public-key signcryption, certificateless signcryption is intrinsically multi-user: there is no distinction between the simpler one-to-one scenario and a fully-fledged multi-user scenario, as user's identities are explicitly associated with ciphertexts.

In defining the security of certificateless signcryption we follow the common approach in the literature [34, 26] where one does not consider attacks targeting signcryptions where the sender and receiver identities are the same. In particular we disallow such queries to relevant oracles and do not accept this type of signcryption as a valid forgery. We discuss this issue further in Section 8.4.

The security game that captures the confidentiality requirement is based on the concept of ciphertext indistinguishability, and is defined as follows:

IND-atk-x

1. $(\mathtt{Msk}, I) \leftarrow \mathtt{Setup}(1^\kappa)$
2. $(\mathtt{m}_0, \mathtt{m}_1, \mathtt{ID}_S^*, \mathtt{ID}_R^*, \mathrm{st}) \leftarrow A_1^{\mathcal{O}_1}(I, \mathtt{aux})$
3. $b \leftarrow \{0,1\}$
4. $\mathtt{c}^* \leftarrow \mathtt{Sc}(\mathtt{m}_b, S_S^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, I)$
5. $b' \leftarrow A_2^{\mathcal{O}_2}(\mathtt{c}^*, \mathrm{st})$

$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-atk-x}}(A) := |2\Pr[b' = b] - 1|.$

Here we require $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ to be distinct. Parameter aux is the empty string when $\mathtt{x} = \mathtt{I}$ and it is the KGC's secret key Msk when $\mathtt{x} = \mathtt{II}$. Note, it is possible that the challenger is not aware of the secret value corresponding to $\mathtt{ID}_S^*$, if the associated public key has been replaced. In this case, we require the adversary to provide this value[1]. Here, implicitly, the challenger continues to use Msk whose value could be unknown to the adversary.

The adversary has access to six oracles:

- **Request Public Key**: On input an identity ID, this oracle returns the corresponding public key. If such a key does not yet exist, it is constructed using the Generate-User-Keys algorithm.
- **Replace Public Key**: On input an identity ID and a valid PK, this oracle replaces the public key associated with ID with PK.
- **Extract Partial Private Key**: On input an identity ID, this oracle returns a partial private key D for that ID, generated using the Extract-Partial-Private-Key algorithm.
- **Extract Full Private Key**: On input an identity ID, this oracle returns a full private key S for that identity S. If such a key does not yet exist, it is constructed using the appropriate algorithms. The adversary is not allowed to query this oracle on any identity for which the corresponding public key has been replaced. This restriction is imposed due to the fact that it is unreasonable to expect that the challenger is able to provide a full private key for a user for which it does not know the secret value. Additionally, the adversary is never allowed to call this oracle on the challenge identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$. To capture *insider security*, this restriction applies only to $\mathtt{ID}_R^*$.

[1] It is possible to formulate a strong challenge oracle which does not run in polynomial time as in Section 2.3.6.

- **De-signcrypt**: On input a ciphertext, a sender's identity, and a receiver's identity, this oracle returns the result of running the $\mathtt{Dsc}$ algorithm on the ciphertext, the sender's public parameters, and the receiver's full private key. Note that, it is possible that the challenger is not aware of the receiver's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it [2]. Of course, the adversary is not allowed to query this oracle in the second stage of the game on $\mathtt{c}^*$ under $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$, unless the public key $\mathtt{PK}_S^*$ of the sender *or* that of the receiver $\mathtt{PK}_R^*$ used to signcrypt $\mathtt{m}_b$ has been replaced after the challenge was issued. We also disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$.

Next we describe the additional oracle restrictions in each attack scenario[3]:

Type-I Adversary (IND-oCCA-I): This scenario models an attacker which is a common user of the system and is not in possession of the KGC's secret key. This type of adversary is not allowed to extract the partial private key for $\mathtt{ID}_R^*$ or $\mathtt{ID}_S^*$ if the public key of this identity has been replaced before the challenge ciphertext was issued. To capture insider security (iCCA), this restriction is lifted from $\mathtt{ID}_S^*$.

In our security analysis we use a weaker formulation of insider Type-I adversaries which we refer to as insider Type-I′. Here the adversary is not allowed to extract the partial private key of $\mathtt{ID}_R^*$ at all.

Type-II Adversary (IND-oCCA-II): This scenario models an honest-but-curious KGC, against which we want to preserve confidentiality. For this type of adversary, the partial private key extraction oracle is not necessary, as the adversary can simply generate these keys itself using $\mathtt{Msk}$. Additionally, this type of adversary is not allowed to replace the public key for $\mathtt{ID}_R^*$ or $\mathtt{ID}_S^*$ before the challenge is issued.

[2] Note that, implicitly, the oracle answers continue to use $\mathtt{Msk}$, which could be unknown to the adversary. Once again, a non-polynomial-time de-signcryption oracle could also be formulated.

[3] In all of the following scenarios, if the adversary does not make any decryption queries it is said to be an IND-oCPA/iCPA-x adversary.

To capture insider security (iCCA), this restriction is lifted from $\mathtt{ID}_S^*$.

The following lemma justifies our definition of Type-I′ attackers.

Lemma 8.1. *If a certificateless signcryption scheme is IND-iCCA secure against Type-II and Type-I′ attackers then it is also IND-iCCA secure against Type-I attackers. More precisely, for any PPT Type-I adversary A there exist PPT adversaries B_0 and B_1 in Type-I′ and Type-II models respectively, such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I}}(A) \leq 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I'}}(B_0) + 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(B_1).$$

Proof. The intuition behind the proof is that we guess if the adversary is going to replace the public key for the challenge identity or extract the partial private key for it and, accordingly, we use the adversary to win a Type-I′ or a Type-II security game.

Let A be an insider Type-I adversary against the CLSC scheme. We construct an algorithm B which has non-negligible advantage against the insider Type-I′ or the insider Type-II security game as follows. Algorithm B receives the parameters from the Type-I′ and Type-II security games. It flips a coin c as to guess if A will be replacing the public key for $\mathtt{ID}_R^*$ (case $c = 0$) or extract the partial private key for it (case $c = 1$). If $c = 0$ then B passes the parameters from the Type-I′ game to A and outputs a random bit in the Type-II game, otherwise it passes the parameters from Type-II game and returns a random bit in Type-I′ game. Let us define the following two events:

- R: the event that A chooses to replace the public key of $\mathtt{ID}_R^*$ in the first stage.
- E: the event that A chooses to extract the partial private key of $\mathtt{ID}_R^*$ at some point.

We now describe how algorithm B answers various queries made by A in each case.

Case $c = 0$: Algorithm B answers the request public key, replace public key, partial private key extraction and decryption oracles using his equivalent oracles. When A outputs two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ and two messages, algorithm B also returns these to its own challenge oracle. The simulation fails if A decides to extract the partial private key of $\mathtt{ID}_R^*$ and not replace its public key (event $\neg R \wedge E$). In this case B outputs a random bit and terminates. The second stage of the game is simulated as in the first case. Note that the simulation in the second stage fails if A ever decides to ask for the partial private key of $\mathtt{ID}_R^*$. This query is allowed if A did not replace the public key of $\mathtt{ID}_R^*$ in the first stage (event $\neg R \wedge E$). When A outputs a bit b', algorithm B also outputs this bit as his own guess.

Case $c = 1$: Algorithm B answers the request public key, replace public key, partial private key extraction and decryption oracles using his equivalent oracles. When A outputs two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ and two messages, algorithm B also returns these to its own challenge oracle. The simulation fails if A decided to replace the public key of $\mathtt{ID}_R^*$ and not extract its partial private key (event $R \wedge \neg E$). In this case B outputs a random bit and terminates. The second stage of the game is simulated as in the first case. Note that the simulation in the second stage is perfect. When A outputs a bit b', algorithm B also outputs this bit as his own guess.

We now analyse the probably that algorithm B returns the correct answer in one of the games it plays. Let $b_{\mathtt{I}'}$ and $b_{\mathtt{II}}$ be the hidden bits in the Type-I$'$ and Type-II games respectively. Let also b_1 and b_2 denote the bits B outputs. Note that if the simulation does terminate unexpectedly, c remains hidden from adversary's view. Note also that A is not allowed to provoke the event $R \wedge E$. We have:

$$2\Pr[b_1 = b_{\mathtt{I}'}] = \Pr[b_1 = b_{\mathtt{I}'} | c = 0] + \Pr[b_1 = b_{\mathtt{I}'} | c = 1] = \Pr[b_1 = b_{\mathtt{I}'} | c = 0] + \frac{1}{2}.$$

And similarly:

$$2\Pr[b_2 = b_{\mathrm{II}}] = \Pr[b_2 = b_{\mathrm{II}} | c = 1] + \frac{1}{2}.$$

And hence:

$$\begin{aligned} \mathrm{Adv}^{\mathtt{IND-iCCA-I'}}_{\mathtt{CLSC}}(B) &= |\Pr[b_1 = b_{\mathrm{I}'} | c = 0] - \frac{1}{2}|, \\ \mathrm{Adv}^{\mathtt{IND-iCCA-II}}_{\mathtt{CLSC}}(B) &= |\Pr[b_2 = b_{\mathrm{II}} | c = 1] - \frac{1}{2}|. \end{aligned}$$

Now:

$$\begin{aligned} \Pr[b_1 = b_{\mathrm{I}'} | c = 0] &= \Pr[b_1 = b_{\mathrm{I}'} \wedge E_1 | c = 0] + \Pr[b_1 = b_{\mathrm{I}'} \wedge E_2 | c = 0] + \\ &+ \Pr[b_1 = b_{\mathrm{I}'} \wedge E_3 | c = 0] \\ &= \Pr[b_1 = b_{\mathrm{I}'} \wedge E_1 | c = 0] + 1/2 + \Pr[b_1 = b_{\mathrm{I}'} \wedge E_3 | c = 0], \end{aligned}$$

where $E_1 := R \wedge \neg E$, $E_2 := \neg R \wedge E$ and $E_3 := \neg R \wedge \neg E$. Similarly:

$$\Pr[b_2 = b_{\mathrm{II}} | c = 1] = 1/2 + \Pr[b_2 = b_{\mathrm{II}} \wedge E_2 | c = 1] + \Pr[b_2 = b_{\mathrm{II}} \wedge E_3 | c = 1].$$

Now if we denote by b_{I} the hidden bit in the resulting Type-I game from the above simulation that A is trying to win, we have:

$$\begin{aligned} \Pr[b' = b_{\mathrm{I}} | c = 0] &= \Pr[b_1 = b_{\mathrm{I}'} \wedge E_1 | c = 0] + \Pr[b' = b_{\mathrm{I}'} \wedge E_2 | c = 0] + \\ &+ \Pr[b_1 = b_{\mathrm{I}'} \wedge E_3 | c = 0]. \end{aligned}$$

And also:

$$\begin{aligned} \Pr[b' = b_{\mathrm{I}} | c = 1] &= \Pr[b' = b_{\mathrm{II}} \wedge E_1 | c = 1] + \Pr[b_2 = b_{\mathrm{II}} \wedge E_2 | c = 1] + \\ &+ \Pr[b_2 = b_{\mathrm{II}} \wedge E_3 | c = 1]. \end{aligned}$$

Adding up the previous two equalities we get:

$$\begin{aligned} 2\Pr[b' = b_{\mathrm{I}}] &= \Pr[b_1 = b_{\mathrm{I}'} | c = 0] - \frac{1}{2} + \Pr[b' = b_{\mathrm{I}'} \wedge E_2 | c = 0] + \\ &+ \Pr[b_2 = b_{\mathrm{II}} | c = 1] - \frac{1}{2} + \Pr[b' = b_{\mathrm{II}} \wedge E_1 | c = 1]. \end{aligned}$$

Subtracting 1 from both sides, taking absolute signs and using the equalities derived for the advantage of B above we obtain:

$$\begin{aligned} \mathrm{Adv}^{\mathtt{IND-iCCA-I}}_{\mathtt{CLSC}}(A) &\leq \mathrm{Adv}^{\mathtt{IND-iCCA-I'}}_{\mathtt{CLSC}}(B) + \mathrm{Adv}^{\mathtt{IND-iCCA-II}}_{\mathtt{CLSC}}(B) + \\ &+ |\Pr[b' = b_{\mathrm{I}'} \wedge E_2 | c = 0] - \frac{1}{2}| + \\ &+ |\Pr[b' = b_{\mathrm{II}} \wedge E_1 | c = 1] - \frac{1}{2}|. \end{aligned}$$

It remains to bound the quantities between the absolute signs above. We achieve this by noting that algorithm B has the same probability of success in Type-I$'$ and Type-II games, if this success is achieved using event E_2 or event E_1. This is because c is independent of A's view until event E_1 or event E_2 occurs in each case. Hence:

$$\begin{aligned} \Pr[b' = b_{\mathrm{I}'} \wedge E_2 | c = 0] &= \Pr[b' = b_{\mathrm{II}} \wedge E_2 | c = 1] = \Pr[b_2 = b_{\mathrm{II}} \wedge E_2 | c = 1] \\ &\leq \Pr[b_2 = b_{\mathrm{II}} | c = 1], \\ \Pr[b' = b_{\mathrm{II}} \wedge E_1 | c = 1] &= \Pr[b' = b_{\mathrm{I}'} \wedge E_1 | c = 0] = \Pr[b_1 = b_{\mathrm{I}'} \wedge E_1 | c = 0] \\ &\leq \Pr[b_1 = b_{\mathrm{I}'} | c = 0]. \end{aligned}$$

Therefore:

$$\mathrm{Adv}^{\mathtt{IND-iCCA-I}}_{\mathtt{CLSC}}(A) \leq 2\mathrm{Adv}^{\mathtt{IND-iCCA-I'}}_{\mathtt{CLSC}}(B) + 2\mathrm{Adv}^{\mathtt{IND-iCCA-II}}_{\mathtt{CLSC}}(B).$$

□

The authenticity property required for certificateless signcryption schemes is captured by the following strong existential unforgeability security model.

UF-atk-x

1. $(\mathtt{Msk}, I) \leftarrow \mathtt{Setup}(1^{\kappa})$

2. $(\mathtt{c}^*, \mathtt{ID}_S^*, \mathtt{ID}_R^*) \leftarrow A^{O}(I, \mathtt{aux})$

where $\mathtt{aux}$ is the empty string when $\mathtt{x} = \mathtt{I}$ and it is the master secret key $\mathtt{Msk}$ when $\mathtt{x} = \mathtt{II}$. For the unforgeability game, we define the adversary's advantage as

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{UF-atk-x}}(A) := \Pr[\mathtt{m}^* \neq \perp \wedge (\mathtt{m}^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, \mathtt{c}^*) \notin L],$$

where $\mathtt{m}^* := \mathtt{Dsc}(\mathtt{c}^*, S_R^*, \mathtt{ID}_R^*, \mathtt{PK}_R^*, \mathtt{ID}_S^*, \mathtt{PK}_S^*, I)$ and $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$ should be distinct. We denote by L the list of inputs made to and the corresponding outputs received from the signcryption oracle. This list consists of entries $(\mathtt{m}, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{c})$, where $\mathtt{PK}_S$ and $\mathtt{PK}_R$ are the public keys corresponding to the queried identities at the time the query is placed. A weaker form of unforgeability can be defined by imposing that the signcryption oracle has not been used to obtain a different ciphertext on the same parameters associated with $\mathtt{c}^*$, namely $\mathtt{m}^*$, $\mathtt{ID}_S^*$, $\mathtt{ID}_R^*$ and the associated public keys at the time of A's termination.

The adversary has access to the same oracles as in the confidentiality game as well as an additional signcryption oracle. We describe the differences to the previous security game:

- **Extract Full Private Key**: Same as in the previous game, but an insider adversary is allowed to query this oracle only on $\mathtt{ID}_R^*$, rather than on $\mathtt{ID}_S^*$.

- **De-signcrypt**: There are no restrictions on calls to this oracle, although the adversary should provide the secret value for the receiver, in case the corresponding public key has been replaced. We disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$.

- **Signcrypt**: On input a message, a sender's identity and a receiver's identity, this oracle returns the result of running the signcryption algorithm on the message, the sender's full private key, and the receiver's public parameters. Note that, it is possible that the challenger is not aware of the sender's secret value, if the associated public key has been replaced. In this case, we require the adversary to provide it. We also disallow queries where $\mathtt{ID}_R = \mathtt{ID}_S$.

Various attack scenarios are as follows[4]:

Type-I Adversary (UF-oCMA-I): This type of adversary is not allowed to extract the partial private keys for $\mathtt{ID}_S^*$ or $\mathtt{ID}_R^*$ if the public keys for these identities have been replaced. To capture insider security (iCMA), this restriction is lifted from $\mathtt{ID}_R^*$.

In our security analysis we use a weaker formulation of insider Type-I adversaries which we refer to as insider Type-I$'$. Here the adversary is not allowed to extract the partial private key of $\mathtt{ID}_S^*$ at all.

Type-II Adversary (UF-oCMA-II): This type of adversary is not allowed to replace the encrypt/verify key for $\mathtt{ID}_S^*$ or $\mathtt{ID}_R^*$. To capture insider security (iCMA), this restriction is lifted from $\mathtt{ID}_R^*$.

The following result, which relates the different authenticity adversarial models, is analogous to that presented for confidentially in Lemma 8.1.

Lemma 8.2. *If a certificateless signcryption scheme is UF-iCMA secure against Type-II and Type-I$'$ attackers then it is also UF-iCMA secure against Type-I attackers. More precisely, for any PPT Type-I adversary A there exist PPT adversaries B_0 and B_1 in Type-I$'$ and Type-II models respectively, such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{UF-iCMA-I}}(A) \leq 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{UF-iCMA-I'}}(B_0) + 2\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{UF-iCMA-II}}(B_1).$$

[4]In all of the following scenarios, if the adversary does not make any signcryption queries it is said to be an UF-oNMA/iNMA-x (no message attack) adversary.

8.2 An Efficient Certificateless Signcryption Scheme

Our certificateless signcryption scheme can be seen as an Encrypt-then-Sign construction where randomness is shared between signature and encryption schemes. Our scheme relies on a computational symmetric bilinear group scheme Γ (see Section 2.2.2). We choose four cryptographic hash functions with domain $\{0,1\}^*$. The co-domain of H_1, H_3 and H_4 is G_1 and that of H_2 is $\{0,1\}^\kappa$. We select $\mathtt{Msk}$ uniformly at random from $\mathbb{Z}_p^*$, set $\mathtt{Mpk} := \mathtt{Msk} \cdot P$ and let $I := (\Gamma, \mathtt{Mpk})$. The partial private key extraction algorithm returns $D := \mathtt{Msk} \cdot Q_{\mathtt{ID}}$ where $Q_{\mathtt{ID}} := H_1(\mathtt{ID})$. The user key generation algorithm returns a random element x from $\mathbb{Z}_p^*$ as the secret value, and $\mathtt{PK} := x \cdot P$ as the public key. The full private key is then set to be $S := (x, D)$. Message, ciphertext and randomness spaces are $\{0,1\}^\kappa$, $G_1 \times \{0,1\}^\kappa \times G_1$ and $\mathbb{Z}_p^*$ respectively. Signcryption and de-signcryption are given below.

$\mathtt{Sc}(\mathtt{m}, S_S, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{Mpk})$

- $r \leftarrow \mathbb{Z}_p^*$; $U \leftarrow rP$
- $T \leftarrow e(\mathtt{Mpk}, H_1(\mathtt{ID}_R))^r$
- $h \leftarrow H_2(U, T, r\mathtt{PK}_R, \mathtt{ID}_R, \mathtt{PK}_R)$
- $V \leftarrow \mathtt{m} \oplus h$
- $H \leftarrow H_3(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
- $H' \leftarrow H_4(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
- $(x_S, D_S) \leftarrow S_S$
- $W \leftarrow D_S + rH + x_S H'$
- $\mathtt{c} \leftarrow (U, V, W)$
- Return $\mathtt{c}$

$\mathtt{Dsc}(\mathtt{c}, S_R, \mathtt{ID}_R, \mathtt{PK}_R, \mathtt{ID}_S, \mathtt{PK}_S, \mathtt{Mpk})$

- $(U, V, W) \leftarrow \mathtt{c}$
- $H \leftarrow H_3(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
- $H' \leftarrow H_4(U, V, \mathtt{ID}_S, \mathtt{PK}_S)$
- If $e(\mathtt{Mpk}, H_1(\mathtt{ID}_S))e(U, H) \neq e(\mathtt{PK}_S, -H')e(P, W)$ return $\perp$
- $(x_R, D_R) \leftarrow S_R$
- $T \leftarrow e(D_R, U)$
- $h \leftarrow H_2(U, T, x_R U, \mathtt{ID}_R, \mathtt{PK}_R)$
- $\mathtt{m} \leftarrow V \oplus h$
- Return $\mathtt{m}$

8.3 Security Analysis

We now turn to the security analysis of this scheme. First, we prove the confidentiality of transmitted messages.

Theorem 8.1. *The certificateless signcryption scheme above is IND-iCCA-I/II secure, in the random oracle model, under the assumption that the gap bilinear Diffie–Hellman problem is intractable in the underlying bilinear group.*

This theorem follows from Lemmas 8.1, 8.3 and 8.4.

Lemma 8.3. *Under the GBDH assumption, no PPT attacker A has non-negligible advantage in winning the IND-iCCA-I′ game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, for any such A there exist a PPT algorithm B such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-I'}}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{GBDH}}(B, q_D^2 + 2q_D q_2 + q_2),$$

where $q_T = q_1 + q_X + q_{SK} + 2q_D + 2$. Here q_1, q_2, q_X, q_{SK} and q_D are the maximum number of queries that the adversary could place to H_1, H_2, partial private key extraction, full private key extraction and de-signcryption oracles.

Proof. On receiving the GBDH challenge tuple (Γ, aP, bP, cP), where the generator is P, algorithm B sets $\mathtt{Mpk} := aP$ and $I = (\Gamma, \mathtt{Mpk})$ and passes them on to A. Algorithm B chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, where q_T is as in the statement of the lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$ algorithm B chooses $r \in \mathbb{Z}_p^*$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(i, \mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$. Otherwise, it returns $Q_{\mathtt{ID}_\ell} = bP$ and adds $(\ell, \mathtt{ID}, \perp)$ to L_1. From now on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Extract Partial Private Key Queries: For each new query $\texttt{ID}$, algorithm B calls H_1 on $\texttt{ID}$ and obtains $(i, \texttt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B returns $D = raP$.

Request Public Key Queries: For each query $\texttt{ID}$, algorithm B checks in list L_K, which is initially empty, if there is a tuple $(\texttt{ID}, \texttt{PK}, x)$. If so, then B returns $\texttt{PK}$. Otherwise, B generates a new key pair, updates the list L_K, and returns the public key.

Replace Public Key Queries: On input $(\texttt{ID}, \texttt{PK})$ algorithm B inserts/updates L_K with tuple $(\texttt{ID}, \texttt{PK}, \perp)$.

Extract Full Private Key Queries: For each new query $\texttt{ID}$, algorithm B calls H_1 on $\texttt{ID}$ and obtains $(i, \texttt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B searches L_K for the entry $(\texttt{ID}, \texttt{PK}, x)$, generating a new key pair if this does not exist, and returns (x, raP).

H_3 **Queries**: For each new query $(U, V, \texttt{ID}, \texttt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p^*$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \texttt{ID}, \texttt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p^*$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \texttt{ID}, \texttt{PK})$, algorithm B proceeds as follows:

1. It checks if the decisional bilinear Diffie–Hellman oracle returns 1 when queried with the tuple (aP, bP, cP, T). If this is the case, algorithm B returns T and stops.

2. Algorithm B goes through the list L_2 with entries $(U, \star, R, \texttt{ID}, \texttt{PK}, h)$, for different values of h, such that the decisional bilinear Diffie–Hellman oracle

returns 1 when queried on the tuple (aP, bP, U, T). Note that in this case $\mathtt{ID} = \mathtt{ID}_\ell$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with T).

3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$ algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling the H_1 and request public key oracles. It returns $\perp$ if verification does not succeed.

2. It calculates $R := x'U$, obtaining x' (and hence $\mathtt{PK}'$) from either the adversary or by calling the request public key oracle.

3. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $T = e(rU, \mathtt{Mpk})$, where $(j, \mathtt{ID}', r)$ is obtained by calling H_1 on $\mathtt{ID}'$, and completes de-signcryption in the usual way placing a query on H_2.

4. If $\mathtt{ID}' = \mathtt{ID}_\ell$ then the pairing cannot be calculated. In order to return a consistent answer, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of T, such that the decisional bilinear Diffie–Hellman oracle returns 1 when queried on (aP, bP, U, T). If such an entry exists, the correct pairing value is found and B decrypts using the hash value h.

5. If B reaches this point of execution, it places the entry $(U, \star, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h. The symbol $\star$ denotes an unknown value of pairing. Note that the identity component of all entries with a $\star$ is $\mathtt{ID}_\ell$.

Eventually, A outputs two messages $(\mathtt{m}_0, \mathtt{m}_1)$ and two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$. Algorithm B places a query on H_1 with input $\mathtt{ID}_R^*$. If the index of $\mathtt{ID}_R^*$ is not ℓ, algorithm B fails. Otherwise it proceeds to construct a challenge as follows. It obtains from L_K the public key $\mathtt{PK}$ corresponding to $\mathtt{ID}_S^*$. Then it sets $U^* := cP$, selects a random bit σ and a random hash value h^* and sets $V^* := \mathtt{m}_\sigma \oplus h^*$. The component W^* is set to be $D_S + rH + x_S H' = D_S + tcP + s\mathtt{PK}$ where t is obtained from L_3, s is obtained from L_4 and D_S is calculated by calling H_1 on $\mathtt{ID}_S^*$. Note that signcryption fails if $\mathtt{ID}_S^* = \mathtt{ID}_R^*$ as B cannot know the value of D_S.

In the second stage, A's queries are answered as before. Eventually, A will output its guess as to which message is signcrypted inside the challenge. Since ℓ is independent of adversary's view, and the list L_1 can be easily seen to have at most q_T elements, with probability $1/q_T$ the adversary will output an identity $\mathtt{ID}_\ell$ with index ℓ. If this event occurs, the simulation is perfect unless the adversary queries H_2 on the challenge-related tuple $(U^*, T^*, R^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$. However, since the hash function H_2 is modelled as a random oracle, the adversary will not have any advantage if this tuple does not appear on L_2. Furthermore, if this happens, B will win the game due to the first step in the simulation of H_2. The lemma follows from this observation and the fact that the total number of decisional bilinear Diffie–Hellman oracle calls that B makes is at most $q_D^2 + 2q_D q_2 + q_2$.

□

Lemma 8.4. *Under the CDH assumption in G_1 no PPT adversary A has non-negligible advantage in winning the IND-iCCA-II game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, for any such A there exist a PPT algorithm B such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{IND-iCCA-II}}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{CDH}}(B),$$

where $q_T = q_{PK} + q_{RPK} + q_{SK} + 2q_D + 2$. *Here* q_{PK} *and* q_{RPK} *are the maximum number of queries that the adversary could place to request public key and replace public key oracles and* q_{SK} *and* q_D *are as before.*

Proof. On receiving the CDH challenge tuple (Γ, aP, bP), with generator P, algorithm B generates a master key pair $(\mathtt{Msk}, \mathtt{Mpk})$ and sets $I := (\Gamma, \mathtt{Mpk})$ and passes these on to A. Algorithm B chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, with q_T as in the statement of the lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the non-repeat query $\mathtt{ID}$ algorithm B chooses $r \in \mathbb{Z}_p^*$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(\mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$.

Request Public Key Queries: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$, algorithm B generates a new key pair $(x, \mathtt{PK})$, updates the list L_K with $(i, \mathtt{ID}, x, \mathtt{PK})$. If $i = \ell$ algorithm B returns aP and adds $(\ell, \mathtt{ID}, aP, \perp)$ to L_K. For this query we denote the identity with $\mathtt{ID}_\ell$.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ on the i-th non-repeat identity $\mathtt{ID}$ algorithm B inserts/updates L_K with tuple $(i, \mathtt{ID}, \mathtt{PK}, \perp)$. If $i = \ell$ then B aborts the simulation.

Extract Full Private Key Queries: For each new query $\mathtt{ID}$, B calls request public key on $\mathtt{ID}$ obtaining $(i, \mathtt{ID}, \mathtt{PK}, x)$. If $i = \ell$, algorithm B aborts the simulation. Otherwise, it calls H_1 on $\mathtt{ID}$ and gets $(\mathtt{ID}, r)$. It returns $(x, r\mathtt{Msk}P)$.

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p^*$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random

value s in $\mathbb{Z}_p^*$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If so, B returns R and stops.
2. Algorithm B goes through the list L_2 looking for entries $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, such that $e(U, bP) = e(P, R)$. Note that in this case $\mathtt{ID} = \mathtt{ID}_\ell$ (see step 5 of de-signcryption simulation below). If such a tuple exists, it returns h (and replaces the symbol $\star$ with R).
3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling H_1 and request public key oracles. It returns $\perp$ if the verification does not succeed.
2. It calculates $T = e(U, r'\mathtt{Mpk})$, where $(\mathtt{ID}', r')$ is obtained from H_1
3. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $R := x'U$, where $(j, \mathtt{ID}', \mathtt{PK}', x')$ is obtained by calling the request public key oracle on $\mathtt{ID}'$, and x' is possibly received from the adversary. It completes de-signcryption in the usual way by placing a query on H_2.
4. If $\mathtt{ID}' = \mathtt{ID}_\ell$, the correct value of R cannot be computed. To answer the query consistently, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for

different values of R, such that $e(U, bP) = e(P, R)$. If such an entry exists, the correct value of R is found, and B decrypts using h.

5. If B reaches this point of execution, B places the entry $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h.

Eventually, A outputs two messages $(\mathtt{m}_0, \mathtt{m}_1)$ and two identities $\mathtt{ID}_S^*$ and $\mathtt{ID}_R^*$. Algorithm B queries the request public key oracle on $\mathtt{ID}_R^*$ and gets $(j, \mathtt{ID}_R^*, \mathtt{PK}^*, x^*)$. If $j \neq \ell$, it fails. Otherwise it proceeds to construct a challenge as follows. It obtains the public key $\mathtt{PK}$ for $\mathtt{ID}_S^*$ by calling the request public key oracle. It sets $U^* = bP$, selects a random bit σ and a random hash value h^* and sets $V^* = \mathtt{m}_\sigma \oplus h^*$. The component W^* is set to be $D_S + rH + x_S H' = D_S + tcP + s\mathtt{PK}$ where t is obtained from L_3 and s is obtained from L_4[5].

In the second stage, A's queries are answered as before. Eventually, A will output its guess as to which message is signcrypted inside the challenge. Since ℓ is independent of adversary's view, with probability $1/q_T$ the adversary will output an identity $\mathtt{ID}_R^*$ with index ℓ. If this event occurs, the simulation is perfect unless the challenge-related tuple $(U^*, T^*, R^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$ is queried from H_2. However, since the hash function H_2 is modelled as a random oracle, the adversary will not have any advantage if this entry does not appear on L_2 list and, in this case, B will have won the game due to step 1 of simulation of H_2. The lemma follows from this observation and the fact that the maximum length of the list L_K is q_T, as stated in the lemma.

□

The authenticity (and non-repudiation) property of the scheme in proved next.

Theorem 8.2. *The certificateless signcryption scheme above is UF-iCMA-I/II secure, in the random oracle model, under the modified GDH assumption in G_1 (see*

[5]Note that unlike Lemma 8.3 note that the simulation is possible for $\mathtt{ID}_S^* = \mathtt{ID}_R^*$.

Definition 2.11).

This theorem follows from Lemmas 8.2, 8.5 and 8.6.

Lemma 8.5. *Under the mGDH assumption in G_1 (see Section 2.2.2), no PPT attacker A has non-negligible advantage in winning the UF-iCMA-I′ game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, for any such A there exist a PPT algorithm B such that:*

$$\mathrm{Adv}_{\mathtt{CLSC}}^{\mathtt{UF-iCMA-I'}}(A) \leq q_T \mathrm{Adv}_{\Gamma}^{\mathtt{mGDH}}(B, q_D^2 + 2q_D q_2) + (q_{SC}(q_{SC} + q_D + q_3 + 1) + 2)/2^{\kappa},$$

where $q_T = q_1 + q_X + q_{SK} + 2q_D + 2q_{SC} + 1$. Here q_3 and q_{SC} is the maximum number of queries that the adversary could place to the H_3 and signcryption oracles and q_1, q_X, q_{SK} and q_D are as before.

Proof. To prove this lemma, we construct an algorithm B which uses A to solve the mGDH problem over G_1. Algorithm B receives an mGDH problem instance (Γ, aP, bP), with generator P, it sets $\mathtt{Mpk} := aP$ and provides $I := (\Gamma, \mathtt{Mpk})$ to A. Algorithm B then chooses an index ℓ uniformly at random in $\{1, \ldots, q_T\}$, where q_T is as in the statement of the lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$ algorithm B chooses $r \in \mathbb{Z}_p^*$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(i, \mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$. Otherwise, it returns $Q_{\mathtt{ID}_\ell} = bP$ and adds $(\ell, \mathtt{ID}, \perp)$ to L_1. From now on we denote the ℓ-th non-repeat identity queried to this oracle with $\mathtt{ID}_\ell$.

Extract Partial Private Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B returns $D = raP$.

Request Public Key Queries: For each query $\mathtt{ID}$, algorithm B checks in list L_K,

which is initially empty, if there is a tuple $(\mathtt{ID}, \mathtt{PK}, x)$. If so, then B returns $\mathtt{PK}$. Otherwise, B generates a new key pair, updates the list L_K, and returns the public key.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ algorithm B inserts/updates L_K with tuple $(\mathtt{ID}, \mathtt{PK}, \perp)$.

Extract Full Private Key Queries: For each new query $\mathtt{ID}$, algorithm B calls H_1 on $\mathtt{ID}$ and obtains $(i, \mathtt{ID}, r)$. If $i = \ell$ then B aborts the simulation. Otherwise, B searches L_K for the entry $(\mathtt{ID}, \mathtt{PK}, x)$, generating a new key pair if this does not exist, and returns (x, raP).

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p^*$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p^*$, updates an initially empty list L_4 with the input, s and sP and returns sP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If this is the case, algorithm B returns R and stops.

2. Algorithm B goes through the list L_2 with entries $(U, \star, R, \mathtt{ID}, \mathtt{PK}, h)$, for different values of h, such that the decisional bilinear Diffie–Hellman oracle returns 1 when queried on the tuple (aP, bP, U, T). Note that in this case $\mathtt{ID} = \mathtt{ID}_\ell$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with T).

3. It goes through list L_2 with entries entry $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, for different values of h, such that $e(U, \mathtt{PK}') = e(P, R)$. If such a tuple exists, it returns h (and

replaces the symbol $\star$ with R).

4. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$ algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling the H_1 and request public key oracles. It returns $\perp$ if verification does not succeed.

2. It checks if $\mathtt{ID} = \mathtt{ID}_\ell$ and if this is the case then B can solve the mGDH problem as described below.

3. It calculates $R := x'U$, obtaining x' (and hence $\mathtt{PK}'$) from either the adversary or by calling the request public key oracle.

4. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $T = e(rU, \mathtt{Mpk})$, where $(j, \mathtt{ID}', r)$ is obtained by calling H_1 on $\mathtt{ID}'$, and completes de-signcryption in the usual way placing a query on H_2.

5. If $\mathtt{ID}' = \mathtt{ID}_\ell$ then the pairing cannot be calculated. In order to return a consistent answer, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of T, such that the decisional bilinear Diffie–Hellman oracle returns 1 when queried on (aP, bP, U, T). If such an entry exists, the correct pairing value is found and B decrypts using the hash value h.

6. If B reaches this point of execution, it places the entry $(U, \star, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h. The symbol $\star$ denotes

an unknown value of pairing. Note that the identity component of all entries with a $\star$ is $\mathtt{ID}_\ell$.

Signcryption Queries: For each new query $(\mathtt{m}, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It calls H_1 on $\mathtt{ID}$. If $\mathtt{ID} \neq \mathtt{ID}_\ell$, algorithm B simply signcrypts the message, getting the secret value x_S from the request public key or the adversary if necessary.

2. If $\mathtt{ID} = \mathtt{ID}_\ell$ (and hence $\mathtt{ID}' \neq \mathtt{ID}_\ell$), algorithm B generates two random values $u, v \in \mathbb{Z}_p^*$, sets $U = vaP$, calculates $T = e(U, r'\mathtt{Mpk})$, obtaining $(j, \mathtt{ID}', r')$ by calling H_1 on $\mathtt{ID}'$.

3. It goes through list L_2 looking for an entry $(U, T, R, \mathtt{ID}', \mathtt{PK}', h)$ for some R such that $e(U, \mathtt{PK}') = e(P, R)$, where $\mathtt{PK}'$ is obtained by calling the request public key oracle on $\mathtt{ID}'$. If such an entry exists, it calculates $V = \mathtt{m} \oplus h$. Otherwise it uses a random h and updates the list L_2 with $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$.

4. Then B defines the hash value $H_3(U, V, \mathtt{ID}_\ell, \mathtt{PK})$ as $H = v^{-1}(uP - Q_{\mathtt{ID}_\ell})$, aborting the simulation if such a hash queries has been responded with a different value before. This means that B updates list L_3 with the tuple $(U, V, \mathtt{ID}_\ell, \mathtt{PK}, \perp, H)$. Finally, B sets $W = uaP + s\mathtt{PK}$, where s is the value obtained by querying H_4 on $(U, V, \mathtt{ID}_\ell, \mathtt{PK})$ and returns (U, V, W). Note that this is a valid signcryption.

Eventually, A outputs a signcryption (U^*, V^*, W^*) from sender $\mathtt{ID}_S^*$ to receiver $\mathtt{ID}_R^*$. Algorithm B now calls H_1 on $\mathtt{ID}_S^*$ and checks if $\mathtt{ID}_S^* = \mathtt{ID}_\ell$ and if this is not the case it aborts execution. Otherwise, it obtains $\mathtt{PK}^*$ by calling the request public key oracle on $\mathtt{ID}_S^*$ and retrieves t^* and s^* from lists L_3 and L_4 by querying H_3 and

H_4 on $(U^*, V^*, \mathtt{ID}_\ell, \mathtt{PK}^*)$. Note that if A succeeded, then the verification condition holds:

$$\begin{aligned} e(P, W^*) &= e(\mathtt{Mpk}, Q_{\mathtt{ID}_\ell}) e(U^*, H^*) e(\mathtt{PK}^*, H'^*) \\ e(P, W^*) &= e(aP, bP) e(U^*, t^*P) e(\mathtt{PK}^*, s^*P) \\ e(P, abP) &= e(P, W^* - t^*U^* - s^*\mathtt{PK}^*), \end{aligned}$$

and thus B can recover

$$abP = W^* - t^*U^* - s^*\mathtt{PK}^*.$$

Let us now analyse the probability that B succeeds in solving the mGDH problem instance. For this to happen, the simulation must run until the end of the game, the adversary must pick a specific identity as $\mathtt{ID}_S^*$, and it must query the hash functions H_3 and H_4 to properly construct the forgery. The probability that A is able to produce a forgery without querying both hash functions is upper bounded by $2/2^\kappa$.

The probability that B aborts the simulation is related with the following events:

- A places a partial key extraction on $\mathtt{ID}_\ell$.
- A places a full private key extraction on $\mathtt{ID}_\ell$.
- B wants to simulate a signcryption query and this leads to an inconsistency in the H_3 simulation.

Note that if A places either of the first two fatal queries, then it could not possible use $\mathtt{ID}_\ell$ as the sender identity in the forgery it produces at the end of the game, so we can pinpoint the probability that B does not abort the simulation due to this events and A picks the only useful case for solving mGDH as $1/q_T$. Note that the maximum length of the list L_1 is q_T, as stated in the lemma.

The latter fatal event occurs if B's simulation triggers a collision in its simulation of H_3. Since the maximum size of L_3 is $q_{SC} + q_D + q_3 + 1$, we can upper

bound the probability that this occurs as $q_{SC}(q_{SC}+q_D+q_3+1)/2^{\kappa}$. The result follows by noting that B makes at most $q_D^2+2q_Dq_2$ queries to its decisional bilinear Diffie–Hellman oracle. □

Lemma 8.6. *Under the CDH assumption in G_1, no PPT attacker A has non-negligible advantage in winning the UF-iCMA-II game against the scheme proposed above, when all hash functions are modelled as random oracles. More precisely, for any such A there exist a PPT algorithm B such that:*

$$\mathrm{Adv}^{\mathtt{UF-iCMA-II}}_{\mathtt{CLSC}}(A) \leq q_T \mathrm{Adv}^{\mathtt{CDH}}_{\Gamma}(B) + (q_{SC}(q_{SC}+q_D+q_3+1)+2)/2^{\kappa},$$

where $q_T = q_{PK}+q_{RPK}+q_{SK}+2q_D+2q_{SC}+1$ and various q_*'s are as before.

Proof. To prove this lemma, we construct an algorithm B which uses A to solve the CDH problem over G_1. Algorithm B receives a CDH problem instance (Γ, aP, bP), with generator P, generates a master key pair $(\mathtt{Msk}, \mathtt{Mpk})$, sets $I := (\Gamma, \mathtt{Mpk})$ and provides these to A. Algorithm B then chooses an index ℓ uniformly at random in $\{1,\ldots,q_T\}$, where q_T is as in the statement of the lemma, and answers various oracle queries as follows:

H_1 **Queries**: On the non-repeat query $\mathtt{ID}$ algorithm B chooses $r \in \mathbb{Z}_p^*$ uniformly at random and sets $Q_{\mathtt{ID}} = rP$. It then adds $(\mathtt{ID}, r)$ to a list L_1 which is initially empty and returns $Q_{\mathtt{ID}}$.

Request Public Key Queries: On the i-th non-repeat query $\mathtt{ID}$, if $i \neq \ell$, algorithm B generates a new key pair $(x, \mathtt{PK})$, updates the list L_K with $(i, \mathtt{ID}, x, \mathtt{PK})$. If $i = \ell$ algorithm B returns aP and adds $(\ell, \mathtt{ID}, aP, \perp)$ to L_K. For this query we denote the identity with $\mathtt{ID}_\ell$.

Replace Public Key Queries: On input $(\mathtt{ID}, \mathtt{PK})$ on the i-th non-repeat identity $\mathtt{ID}$ algorithm B inserts/updates L_K with tuple $(i, \mathtt{ID}, \mathtt{PK}, \perp)$. If $i = \ell$ then B aborts the

simulation.

Extract Full Private Key Queries: For each new query $\mathtt{ID}$, B calls request public key on $\mathtt{ID}$ obtaining $(i, \mathtt{ID}, \mathtt{PK}, x)$. If $i = \ell$, algorithm B aborts the simulation. Otherwise, it calls H_1 on $\mathtt{ID}$ and gets $(\mathtt{ID}, r)$. It returns $(x, r\mathtt{Msk}P)$.

H_3 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value t in $\mathbb{Z}_p^*$, updates an initially empty list L_3 with the input, t and tP and returns tP.

H_4 **Queries**: For each new query $(U, V, \mathtt{ID}, \mathtt{PK})$, algorithm B generates a random value s in $\mathbb{Z}_p^*$, updates an initially empty list L_4 with the input, s and sbP and returns sbP.

H_2 **Queries**: For each new query $(U, T, R, \mathtt{ID}, \mathtt{PK})$, algorithm B proceeds as follows:

1. It checks if $e(aP, bP) = e(P, R)$. If so, B returns R and stops.
2. Algorithm B goes through the list L_2 looking for entries $(U, T, \star, \mathtt{ID}, \mathtt{PK}, h)$, such that $e(U, \mathtt{PK}) = e(P, R)$. If such a tuple exists, it returns h (and replaces the symbol $\star$ with R).
3. If B reaches this point of execution, it returns a random h and updates the list L_2, which is initially empty, with a tuple containing the input and return values.

De-signcryption Queries: For each new query $(U, V, W, \mathtt{ID}, \mathtt{ID}')$, algorithm B proceeds as follows:

1. It executes the verification part of the de-signcryption algorithm obtaining $Q_{\mathtt{ID}}$ and $\mathtt{PK}$ by calling H_1 and request public key oracles. It returns $\perp$ if the verification does not succeed.

2. It checks if $\mathtt{ID} = \mathtt{ID}_\ell$ and if this is the case then B can solve the mGDH problem as described below.

3. It calculates $T = e(U, r'\mathtt{Mpk})$, where $(\mathtt{ID}', r')$ is obtained from H_1

4. If $\mathtt{ID}' \neq \mathtt{ID}_\ell$, it calculates $R := x'U$, where $(j, \mathtt{ID}', \mathtt{PK}', x')$ is obtained by calling the request public key oracle on $\mathtt{ID}'$, and x' is possibly received from the adversary. It completes de-signcryption in the usual way by placing a query on H_2.

5. If $\mathtt{ID}' = \mathtt{ID}_\ell$, the correct value of R cannot be computed. To answer the query consistently, B goes through L_2 and looks for a tuple $(U, T, R, \mathtt{ID}_\ell, \mathtt{PK}', h)$, for different values of R, such that $e(U, bP) = e(P, R)$. If such an entry exists, the correct value of R is found, and B decrypts using h.

6. If B reaches this point of execution, B places the entry $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$ for a random h on list L_2 and decrypts using this h.

Signcryption Queries: For each new query $(\mathtt{m}, \mathtt{ID}, \mathtt{ID}')$, algorithm B calls the request public key oracle on $\mathtt{ID}$ and proceeds as follows:

1. If $\mathtt{ID} \neq \mathtt{ID}_\ell$, algorithm B simply signcrypts the message, getting the secret value x_S from the request public key or the adversary if necessary.

2. If $\mathtt{ID} = \mathtt{ID}_\ell$ algorithm B generates two random values $u, v \in \mathbb{Z}_p^*$, sets $U = vaP$ and calculates $T = e(U, \mathtt{Msk}Q_{\mathtt{ID}_j})$.

3. It goes through list L_2 looking for an entry $(U, T, R, \mathtt{ID}', \mathtt{PK}', h)$ for some R such that $e(U, \mathtt{PK}') = e(P, R)$, where $\mathtt{PK}'$ is obtained by calling the request public key oracle on $\mathtt{ID}'$. If such an entry exists, it calculates $V = \mathtt{m} \oplus h$. Otherwise it uses a random h and updates the list L_2 with $(U, T, \star, \mathtt{ID}', \mathtt{PK}', h)$.

4. Then B defines the hash value $H_3(U,V,\mathtt{ID}_\ell,\mathtt{PK})$ as $H = v^{-1}(uP - H_4)$, aborting the simulation if such a hash response has been given before, where H_4 is the output of $H_4(U,V,\mathtt{ID}_\ell,\mathtt{PK})$. This means that B updates list L_3 with tuple $(U,V,\mathtt{ID}_\ell,\mathtt{PK},\perp,H)$. Finally, B sets $W = D_S + uaP$ and returns (U,V,W). Note that this is a valid signcryption.

Eventually, A outputs a valid signcryption (U^*,V^*,W^*) from sender $\mathtt{ID}_S^*$ to receiver $\mathtt{ID}_R^*$. Algorithm B now calls the request public key oracle on $\mathtt{ID}_S^*$, obtains $\mathtt{PK}^*$, and checks if $\mathtt{ID}_S^* = \mathtt{ID}_\ell$. If this is not the case it aborts the execution. Otherwise, it retrieves t^* and s^* from lists L_3 and L_4 by querying H_3 and H_4 on $(U^*,V^*,\mathtt{ID}_\ell,\mathtt{PK}^*)$. Note that if A succeeded, then the verification condition holds:

$$\begin{aligned} e(P,W^*) &= e(\mathtt{Mpk},Q_{\mathtt{ID}_\ell})e(U^*,H^*)e(\mathtt{PK}^*,H'^*) \\ e(P,W^*) &= e(\mathtt{Mpk},Q_{\mathtt{ID}_\ell})e(U^*,t^*P)e(aP,s^*bP) \\ e(P,s^*abP) &= e(P,W^* - D_{\mathtt{ID}_\ell} - t^*U^*), \end{aligned}$$

and thus B can recover

$$abP = (W^* - D_{\mathtt{ID}_\ell} - t^*U^*)/s^*.$$

Let us now analyse the probability that B succeeds in solving CDH. For this to happen, the simulation must run until the end of the game, the adversary must pick a specific identity as $\mathtt{ID}_S^*$, and it must query the hash functions H_3 and H_4 to properly construct the forgery. The probability that A is able to produce a forgery without querying both hash functions is upper bounded by $2/2^\kappa$. The probability that B aborts the simulation is related with the following events:

- A places a full private key extraction on $\mathtt{ID}_\ell$.
- B wants to simulate a signcryption query and this leads to an inconsistency

in the H_3 simulation.

Note that if A places the first fatal query, then it could not possible use $\texttt{ID}_\ell$ as the sender identity in the forgery it produces at the end of the game, so we can pinpoint the probability that B does not abort the simulation due to this events and A picks the only useful case for solving CDH as $1/q_T$. Note that the maximum length of the list L_K is q_T, as stated in the lemma.

The latter fatal event occurs if B's simulation triggers a collision in its simulation of H_3. Since the maximum size of L_3 is $q_{SC}+q_D+q_3+1$, we can upper bound the probability that this occurs as $q_{SC}(q_{SC}+q_D+q_3+1)/2^{\kappa}$. The result follows.

□

8.4 Discussion

The security proof for the certificateless signcryption scheme presented in the previous section has several interesting aspects which we will now discuss.

Full Domain Hash: We chose not to adopt Coron's technique [36] to obtain tighter security reductions in the analysis of authenticity for the sake of clarity in the proof presentation. Adaptation of this technique to the certificateless signcryption case can be achieved following the strategy introduced by Libert and Quisquater [64] for identity-based signature schemes. However, it is important to emphasise an issue specific to the certificateless setting which renders this adaptation less straightforward.

The adaptive power of a Type-I attacker as defined in [5] allows the attacker to decide whether it replaces the public key for the challenge identity or it extracts the associated partial private key. This means that a direct adaptation of the proof in [64], which embeds the hard problem instance in a fraction of the partial private keys which arise in the security game, is meaningless for Type-I adversaries that

extract the partial private key for the challenge identity.

This observation motivated the definition of the Type-I′ attack model and Lemmas 8.1 and 8.2 relating Type-I and Type-II security with this new variant. The limited adaptive behaviour of Type-I′ attackers permits applying Coron's technique directly in the certificateless scenario. As an example of why this is a relevant contribution, we refer to the certificateless signature proposed in [90], which is claimed to be secure against Type-I attackers. The proof which is presented for this scheme is flawed, and actually establishes security against more limited Type-I′ adversaries.

Randomness Reuse: The proposed signcryption scheme is structured internally as an Encrypt-then-Sign construction using algorithms from [35] and [90] and sharing randomness between the two schemes. The encryption algorithm can be shown to be IND-CPA secure, whereas the signature algorithm is UF-CMA secure. The expected security of our construction, which follows from the work of An et al. [6], is therefore IND-CCA security against outsider adversaries and full insider UF-CMA security. It is interesting to note, however, that our scheme presents full insider security for confidentiality. This is due to the reuse of randomness between the encryption and signature components which intuitively prevents an insider adversary from being able to forge a valid signcryption from another one for which it does not know the implicit randomness.

We are able to save a few exponentiations and one ciphertext element through the randomness-reuse technique discussed in Chapter 5. Efficiency benefits also justify our choice of the GBDH problem in the security reduction. The gap oracle allows us to construct a consistent simulation without resorting to a generic transformation akin to that in [63] which would add an extra ciphertext element to the scheme and a costly consistency check in de-signcryption. As a final note on the efficiency of the scheme, we note that we could have based our construction on the

certificateless encryption scheme in [5]. This would provide a small computational gain if one considered public key validity check could be pre-computed. However, this would imply reducing the scheme's security to a less standard variant of the GBDH problem used in [5].

Self-Signcryption: We note that, although our security models disallow attacks targeting signcryptions where the sender and receiver identities are the same, it is possible to modify our proof of security to account for this type of attacks. However, one would need less standard versions of the underlying hard problems to construct the security reduction. It is arguable whether this sort of security property is relevant in practice, although specific applications such as protecting one's files or previously sent encrypted e-mails may be used to justify it.

Malicious KGCs: Malicious KGC attacks have not been considered in our work. However, the proposed scheme withstands the restricted attacks described in [7], which consist of allowing a malicious KGC to generate the $(\mathtt{Msk}, I)$ pair (excluding the computational group scheme) itself as long as it provides these to the challenger. As described in Section 2.3.6, a more realistic and stronger malicious KGC security model would only require that the adversary outputs the public parameters I. We leave it as an open problem to find a certificateless signcryption scheme which can be proven secure in this stronger security model.

Bibliography

[1] M. Abe, Y. Cui, H. Imai and E. Kiltz. Efficient Hybrid Encryption from ID-Based Encryption. *Cryptology ePrint Archive*, Report 2007/023. 2007.

[2] S.S. Al-Riyami. *Cryptographic Schemes Based on Elliptic Curve Pairings*, Ph.D. Thesis, University of London, 2004.

[3] S.S. Al-Riyami, J. Malone-Lee and N.P. Smart. Escrow-Free Encryption Supporting Cryptographic Workflow. *International Journal of Information Security*, Vol. 5, No. 4. October 2006.

[4] S.S. Al-Riyami and K.G. Paterson. CBE from CL-PKE: A Generic Construction and Efficient Schemes. *Public Key Cryptography – PKC 2005*, LNCS 3386:398–415. Springer-Verlag, 2005.

[5] S.S. Al-Riyami and K.G. Paterson. Certificateless Public-Key Cryptography. *Advances in Cryptology – ASIACRYPT 2003*, LNCS 2894:452–473. Springer-Verlag, 2003.

[6] J.H. An, Y. Dodis and T. Rabin. On the Security of Joint Signature and Encryption. *Advances in Cryptology – EUROCRYPT 2002*, LNCS 2332:83–107. Springer-Verlag, 2002.

[7] M.H. Au, J. Chen, J.K. Liu, Y. Mu, D.S. Wong and G. Yang. Malicious KGC Attacks in Certificateless Cryptography. *2007 ACM Symposium on*

Information, Computer and Communications Security, pp. 302–311. March 2007.

[8] M. Barbosa and P. Farshim. Certificateless Signcryption. *ACM Symposium on Information, Computer and Communications Security – ASIACCS 2008*, 2008.

[9] M. Barbosa and P. Farshim. Efficient Identity-Based Key Encapsulation to Multiple Parties. *Cryptography and Coding*, LNCS 3796:428–441. Springer-Verlag, 2005.

[10] M. Barbosa and P. Farshim. Randomness Reuse: Improvements and Extensions. *Cryptography and Coding*, LNCS 4887:261–280. Springer-Verlag, 2007.

[11] M. Barbosa and P. Farshim. Secure Cryptographic Workflow in the Standard Model. *Progress in Cryptology – INDOCRYPT 2006*, LNCS 4329:379–393. Springer-Verlag, 2006.

[12] M. Bellare. Practice-Oriented Provable-Security. *Proceedings of First International Workshop on Information Security*, LNCS 1396:221–231. Springer-Verlag, 1998.

[13] M. Bellare, A. Boldyreva and S. Micali. Public-Key Encryption in a Multi-User Setting: Security Proofs and Improvements. *Advances in Cryptology – EUROCRYPT 2000*, LNCS 1807:259–274. Springer-Verlag, 2000.

[14] M. Bellare, A. Boldyreva and J. Staddon. Randomness Re-Use in Multi-Recipient Encryption Schemes. *Public Key Cryptography – PKC 2003*, LNCS 2567:85–99. Springer-Verlag, 2003.

[15] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Advances in Cryptology – CRYPTO '98*, LNCS 1462:26–45. Springer-Verlag, 1998.

[16] M. Bellare, T. Kohno and V. Shoup. Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation. *Third ACM Conference on Computer and Communications Security – CCS*, 2006.

[17] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. *Proceedings of the First Annual Conference on Computer and Communications Security*, pp. 62–73. 1993.

[18] M. Bellare and P. Rogaway. The Game-Playing Technique. *Cryptology ePrint Archive*, Report 2004/331. 2004.

[19] J. Benaloh and J. Leichter. Generalized Secret Sharing and Monotone Functions. *Advances in Cryptology – CRYPTO '88*, LNCS 403:27–35. Springer-Verlag, 1990.

[20] K. Bentahar, P. Farshim, J. Malone-Lee and N.P. Smart. Generic Constructions of Identity-Based and Certificateless KEMs. *Cryptology ePrint Archive*, Report 2005/058. 2005.

[21] T.E. Bjørstad and A.W. Dent. Building Better Signcryption Schemes with Tag-KEMs. *Public Key Cryptography – PKC 2006*, LNCS 3958:491–507, Springer-Verlag 2006.

[22] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. *Advances in Cryptology – EUROCRYPT 2004*, LNCS 3027:223–238. Springer-Verlag, 2004.

[23] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32:586–615. 2003.

[24] D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. *Proceedings of RSA-CT 2005*, LNCS 3376:87–103. Springer-Verlag, 2005.

[25] D. Boneh, B. Lynn and H. Shacham. Short Signatures from the Weil Pairing. *Advances in Cryptology – ASIACRYPT 2001*, LNCS 2248:514–532. Springer-Verlag, 2001.

[26] X. Boyen. Multipurpose Identity-Based Signcryption: A Swiss Army Knife for Identity-Based Cryptography. *Advances in Cryptology – CRYPTO 2003*, LNCS 2729:383–399. Springer-Verlag, 2003.

[27] R.W. Bradshaw, J.E. Holt and K.E. Seamons. Concealing Complex Policies with Hidden Credentials. *Eleventh ACM Conference on Computer and Communications Security*, pp. 146–157. 2004.

[28] R. Canetti, O. Goldreich and S. Halevi. The Random Oracle Methodology, Revisited. *Journal of the ACM (JACM)*, Vol. 51, No. 4, pp. 557–594. 2004.

[29] R. Canetti, S. Halevi and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *Advances in Cryptology – EUROCRYPT 2004*, LNCS 3027:207–222. Springer-Verlag, 2004.

[30] L. Chen and Z. Cheng. Security Proof of Sakai–Kasahara's Identity-Based Encryption Scheme. *Cryptography and Coding*, LNCS 3796:442–459. Springer-Verlag, 2005.

[31] L. Chen, Z. Cheng, J. Malone-Lee and N.P. Smart. An Efficient ID-KEM Based on the Sakai–Kasahara Key Construction. *IEE Proceedings Information Security*, Vol. 153, pp. 19–26, 2006.

[32] L. Chen and K. Harrison. Multiple Trusted Authorities in Identifier Based Cryptography from Pairings on Elliptic Curves. *Technical Report, HPL-2003-48*, HP Laboratories, 2003.

[33] L. Chen, K. Harrison, D. Soldera and N.P. Smart. Applications of Multiple Trusted Authorities in Pairing Based Cryptosystems. *Proceedings InfraSec 2002*, LNCS 2437:260–275. Springer-Verlag, 2002.

[34] L. Chen and J. Malone-Lee. Improved Identity-Based Signcryption. *Public Key Cryptography – PKC 2005*, LNCS 3386:362–379. Springer-Verlag, 2005.

[35] Z. Cheng and R. Comley. Efficient Certificateless Public Key Encryption. *Cryptology ePrint Archive*, Report 2005/012. 2005.

[36] J.-S. Coron. On the Exact Security of Full Domain Hash. *Advances in Cryptology – CRYPTO 2000*, LNCS 1880:229–235. Springer-Verlag, 2000.

[37] R. Cramer and V. Shoup. A Practical Public-Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. *Advances in Cryptology – CRYPTO '98*, LNCS 1462:13–25. Springer-Verlag, 1998.

[38] A.W. Dent. A Designer's Guide to KEMs. *Cryptography and Coding*, LNCS 2898:133–151. Springer-Verlag, 2003.

[39] A.W. Dent. A Survey of Certificateless Encryption Schemes and Security Models. *Cryptology ePrint Archive*, Report 2006/211. 2006.

[40] A.W. Dent. *Private Communication*. 2008.

[41] A.W. Dent, B. Libert and K.G. Paterson. Certificateless Encryption Schemes Strongly Secure in the Standard Model. *Cryptology ePrint Archive*, Report 2007/121. 2007.

[42] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[43] Y. Dodis and J. Katz. Chosen-Ciphertext Security of Multiple Encryption. *Theory of Cryptography – TCC 2005*, LNCS 3378:188–209. Springer-Verlag, 2005.

[44] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *Advances in Cryptology – CRYPTO '84*, LNCS 196:10–18. Springer-Verlag, 1985.

[45] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Advances in Cryptology – CRYPTO '99*, LNCS 1666:537–554. Springer-Verlag, 1999.

[46] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA–OAEP Is Secure under the RSA Assumption. *Advances in Cryptology – CRYPTO 2001*, LNCS 2139:260–274. Springer-Verlag, 2001.

[47] D. Galindo, S.M. Molleví, P. Morillo and J.L. Villar. Fujisaki–Okamoto Hybrid Encryption Revisited. *International Journal of Information Security*, 4(4):228–241. Springer-Verlag, 2005.

[48] C. Gentry. Practical Identity-Based Encryption without Random Oracles. *Advances in Cryptology – EUROCRYPT 2006*, LNCS 4004:445–464. Springer-Verlag, 2006.

[49] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and Systems Sciences*, 28:270–299. 1984.

[50] V. Goyal, O. Pandey, A. Sahai and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. *ACM Conference on Computer and Communications Security*, pp. 89–98, 2006.

[51] J. Hastad. Solving Simultaneous Modular Equations of Low Degree. *SIMA Journal on Computing*, Vol. 17, No. 2, pp. 336–341. April 1988.

[52] J. Herranz and D. Hofheinz and E. Kiltz. KEM/DEM: Necessary and Sufficient Conditions for Secure Hybrid Encryption. *Cryptology ePrint Archive*, Report 2006/265. 2006.

[53] J.E. Holt, R.W. Bradshaw, K.E. Seamons and H. Orman. Hidden Credentials. *2nd ACM Workshop on Privacy in the Electronic Society*, pp. 1–8, 2003.

[54] B.C. Hu, D.S. Wong, Z. Zhang and X. Deng. Certificateless Signature: A New Security Model and an Improved Generic Construction. *Designs, Codes and Cryptography*, Vol. 42, Issue 2, pp. 109–126. Kluwer Academic Publishers, 2007.

[55] X. Huang, W. Susilo, Y. Mu and F. Zhang. On the Security of Certificateless Signature Schemes from Asiacrypt 2003. *Cryptology and Network Security – CANS 2005*, LNCS 3810:13–25. Springer-Verlag, 2005.

[56] M. Joye, J. Quisquater and M. Yung. On the Power of Misbehaving Adversaries and Security Analysis of the Original EPOC. *Topics in Cryptography – CT-RSA 2001*, LNCS 2020:208–222. Springer-Verlag, 2001.

[57] J. Katz and M. Yung. Characterization of Security Notions for Probabilistic Private-Key Encryption. *Journal of Cryptology*, 19(1):67–95. Springer-Verlag, 2005.

[58] E. Kiltz. Chosen-Ciphertext Secure Identity-Based Encryption in the Standard Model with Short Ciphertexts. *Cryptology ePrint Archive*, Report 2006/122. 2006.

[59] H. Krawczyk. Secret Sharing Made Short. *Advances in Cryptology – CRYPTO '93*, LNCS 773:136–146. Springer-Verlag, 1994.

[60] K. Kurosawa. Multi-Recipient Public-Key Encryption with Shortened Ciphertext. *Public Key Cryptography – PKC 2002*, LNCS 2274:48–63. Springer-Verlag, 2002.

[61] L. Lamport. Constructing Digital Signatures from a One-Way Function. *Technical Report CSL-98*, SRI International, 1979.

[62] B. Libert and J.-J. Quisquater. New Identity-Based Signcryption Schemes from Pairings. *IEEE Information Theory Workshop 2003*, pp. 155–158, January 2003.

[63] B. Libert and J.-J. Quisquater. On Constructing Certificateless Cryptosystems from Identity Based Encryption. *Public Key Cryptography – PKC 2006*, LNCS 3958:474–490. Springer-Verlag, 2006.

[64] B. Libert and J.-J. Quisquater. The Exact Security of an Identity Based Signature and its Applications. *Cryptology ePrint Archive*, Report 2004/102. 2004.

[65] Y. Lindell. A Simpler Construction of CCA2-Secure Public-Key Encryption under General Assumptions. *Journal of Cryptology*, 19(3):359–377. Springer-Verlag, 2006.

[66] B. Lynn. Authenticated Identity-Based Encryption. *Cryptology ePrint Archive*, Report 2002/072. 2002.

[67] J. Malone-Lee. Identity-Based Signcryption. *Cryptology ePrint Archive*, Report 2002/098. 2002.

[68] J. Malone-Lee. Signcryption with Non-Interactive Non-Repudiation. *Designs, Codes and Cryptography*, 37(1):81–109. October 2005.

[69] A. Miyaji, M. Nakabayashi and S. Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A:1234–1243. 2001.

[70] W. Nagao, Y. Manabe and T. Okamoto. On the Equivalence of Several Security Notions of Key Encapsulation Mechanism. *Cryptology ePrint Archive*, Report 2006/268. 2006.

[71] T. Okamoto and D. Pointcheval. The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes. *Public Key Cryptography – PKC 2001*, LNCS 1992:104–118. Springer-Verlag, 2001.

[72] K.G. Paterson. Cryptography from Pairings: A Snapshot of Current Research. *Information Security Technical Report*, 7:41–54, 2002.

[73] J. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32:918–924, 1978.

[74] M. Rabin. Digitized Signatures and Public Key functions as Intractable as Factorization. *MIT/LCS/TR-212*, MIT Laboratory for Computer Science, 1979.

[75] R.L. Rivest, A. Shamir and L.M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126. February 1978.

[76] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. *Advances in Cryptology – EUROCRYPT 2005*, LNCS 3494:457–473. Springer-Verlag, 2005.

[77] R. Sakai and M. Kasahara. ID-Based Cryptosystems with Pairing on Elliptic Curve. *2003 Symposium on Cryptography and Information Security – SCIS 2003*, 2003.

[78] O. Schirokauer. Using Number Fields to Compute Logarithms in Finite Fields. *Mathematics of Computation*, 69:1267–1283, 2000.

[79] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, 1979.

[80] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. *Advances in Cryptology – CRYPTO '84*, LNCS 196:47–53. Springer-Verlag, 1985.

[81] C. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, Vol. 28, No. 2, pp. 656–715, 1949.

[82] N.P. Smart. Access Control Using Pairing Based Cryptography. *Topics in Cryptology – CT-RSA 2003*, LNCS 2612:111–121. Springer-Verlag, 2003.

[83] N.P. Smart. Efficient Key Encapsulation to Multiple Parties. *Security in Communication Networks*, LNCS 3352:208–219. Springer-Verlag, 2005.

[84] Y. Shi and J. Li. Provable Efficient Certificateless Public Key Encryption. *Cryptology ePrint Archive*, Report 2005/287. 2005.

[85] V. Shoup. *A Computational Introduction to Number Theory and Algebra*, Cambridge University Press, 2005. ISBN-10: 0521851548.

[86] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption. *Cryptology ePrint Archive*, Report 2001/112. 2002.

[87] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. *Cryptology ePrint Archive*, Report 2004/332. 2004.

[88] B. Waters. Efficient Identity-Based Encryption Without Random Oracles. *Advances in Cryptology – EUROCRYPT 2005*, LNCS 3494:114–127. Springer-Verlag, 2005.

[89] P. Yang, T. Kitagawa, G. Hanaoka, R. Zhang, K. Matsuura and H. Imai. Applying Fujisaki-Okamoto to Identity-Based Encryption. *16th Applied Algebra, Algebraic Algorithms, and Error Correcting Codes – AAECC 2006*, LNCS 3857:183–192. Springer-Verlag, 2006.

[90] Z. Zhang, D.S. Wong, J. Xu and D. Feng. Certificateless Public-Key Signature: Security Model and Efficient Construction. *4th International Conference on Applied Cryptography and Network Security – ACNS 2006*, LNCS 3989:293–308. Springer-Verlag, 2006.

[91] Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption). *Advances in Cryptology – CRYPTO '97*, LNCS 1294:165–179, Springer-Verlag, 1997.