IMPROVEMENTS AND GENERALISATIONS OF SIGNCRYPTION SCHEMES

Wei Zhang

Royal Holloway and Bedford New College,
University of London

Thesis submitted to

The University of London

for the degree of

Doctor of Philosophy

2013.

to my parents

Declaration of Authorship

I, Wei Zhang, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed: Wei Zhang

Date: 28 February 2013

Abstract

In this work, we study the cryptographic primitive: signcryption, which combines the functionalities of digital signatures and public-key encryption.

We first propose two generic transforms from meta-ElGamal signature schemes to signcryption schemes. These constructions can be thought of as generalisations of the signcryption schemes by Zheng and Gamage *et al.* Our results show that a large class of signcryption schemes are outsider IND-CCA2 secure and insider UF-CMA secure. As a by-product, we also show that the meta-ElGamal signature schemes, for which no previous formal security proofs have been shown, are UF-CMA secure.

We then propose a modification of one of the transforms in order to achieve insider IND-CCA2 security in addition to insider UF-CMA security. This modification costs just one extra exponential operation. In particular, we can apply this modification to the Zheng signcryption scheme to make it fully insider secure.

Finally, we propose a generic transform from a two-key signcryption scheme to a one-key signcryption scheme while preserving both confidentiality and unforgeability. Our result shows that if we have an insider IND-CCA2 and UF-CMA secure two-key signcryption scheme, then it can be turned into an insider IND-CCA2 and UF-CMA secure one-key signcryption scheme. We also show that an insider IND-CCA2 and UF-CMA secure one-key signcryption scheme induces a secure combined public-key scheme; that is, a combination of a signature scheme and a public-key encryption scheme that can securely share the same key pair. Combining previous results suggests that we can obtain a large class of insider secure one-key signcryption schemes from meta-ElGamal signature schemes, and that each of them can induce a secure combined public-key scheme.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my two supervisors: Dr. Alexander W. Dent and Prof. Keith Martin.

Alex accepted me as a beginner in cryptography, and patiently guided me throughout the research. His insightful vision and expertise shined a path towards this thesis.

As Alex left for his new career, Keith voluntarily took the responsibility to help me to go through the writing-up stage. His rigour, patience and efficiency played a key role in shaping and finalising my thesis.

Without Alex's guidance and support, it would have not been possible to start the thesis. Without Keith's encouragement and help, it would have not been possible to complete the thesis.

I am also very grateful to Dr. Cid Carlos for offering me a post doctoral position while I was still writing up. His consideration and thoughtfulness helped me to finish this thesis.

I would also like to thank Liz and Qin for making my social life during the PhD study as pleasant as the undergraduate, Gaven for teaching me juggling and improving my juggling skills, and members of Crypto-House: Amy, Anastasia, Ciaran and Gary, for being nice housemates, colleagues and friends.

Last but not least, I would like to thank the ISG for providing such a great environment to make my stay a most rewarding and memorable one.

Contents

Chapter 1

Introduction

Signcryption is a cryptographic primitive that offers the functionalities of public-key encryption and digital signatures at the same time. Before introducing signcryption, we first provide a basic introduction to cryptography and then introduce the concept of provable security, which is the main tool for security analysis of cryptographic schemes in this thesis. An outline of the thesis will be provided following the introduction to signcryption.

1.1 Cryptography

Cryptography means, literally, the art of secret writing. The contemporary analogy of this is that modern cryptography protects computer communications from eavesdroppers. In fact, modern cryptography provides mathematical techniques for establishing a wide range of different tools to protect electronic information, not just hiding it from unauthorised parties.

Most cryptographic protocols take place between two players, often referred to as Alice and Bob, who would like to communicate with each other securely. The main tools of cryptography include, but are not limited to, those that provide confidentiality, integrity and authentication of communications between Alice and Bob. More precisely:

- *Confidentiality* is to ensure that no-one can read a message except the intended receiver Bob.

- *Integrity* is to assure the receiver Bob that the received data has not been altered in any way from the original message created by Alice.

- *Authentication* is a general term which includes *data origin authentication* and *entity authentication*; data origin authentication (message authentication) is to provide Bob with assurance that the origin of the data is Alice, as claimed; entity authentication (identification) is to provide assurance that Bob is communicating with Alice, as claimed.

Confidentiality is often achieved by *encryption*, which converts a *plaintext* into a *ciphertext*. Encryption can be classified either as *symmetric* encryption or *asymmetric* encryption, also known as *public-key* encryption.

Symmetric encryption requires a secret key that is securely pre-shared between Alice and Bob. The secret key is used for both encrypting and decrypting. It is sometimes impractical for Alice and Bob to securely pre-share a secret key. Public-key encryption overcomes this difficulty. In the public-key setting, Bob has a pair of keys, a private key, which is only known to Bob, and a public key, which is known to everyone including Alice. If Alice wants to send a message to Bob, she encrypts the message with Bob's public key, and sends the ciphertext to Bob. When Bob receives the ciphertext, he decrypts the ciphertext using his private key.

It is worth noting that public-key encryption generally involves more computational and communication overheads than symmetric encryption.

Integrity and data origin authentication are often achieved by *digital signatures*. Alice has a pair of keys, a private key and a public key. When Alice wishes to sign a message for Bob, she signs the message with her private key, and sends the signature to Bob. When Bob receives the signature, he verifies the signature using Alice's public key. Successful verification implies that the message is indeed sent from Alice and no alterations were made to the message.

Public-key encryption and digital signatures are just two examples of many *cryptographic primitives*. Another well-recognised cryptographic primitive is a *cryptographic hash function*. A hash function is an algorithm that takes strings of arbitrary length and returns a fixed-length string, known as the *hash value*, such that any change to the input will result in a change to the hash value. The hash value can sometimes be used to provide a way of verifying the integrity of the input. There are three basic requirements of a cryptographic hash function:

- It is infeasible to recover the input from its hash value.
- Given an input and its hash value, it is infeasible to find a different input with the same hash value.
- It is infeasible to find two inputs that have the same hash value.

A formal and detailed introduction to cryptography can be found in [24].

1.2 Provable Security

The apparently simple question of whether a particular cryptographic scheme is "secure" is surprisingly hard to answer. In the old days, the test of a new cryptographic scheme was often to use it until someone broke it. The scheme was then either discarded or modified and used again until the next breakage. This process does not seem appropriate for today's information-based world, where the ability to manipulate information has greater consequences and hence the necessity of protecting information is more important.

This is why mathematical techniques have been adapted to bring rigour and precision to the analysis of cryptographic schemes. We can now describe a cryptographic scheme mathematically, define its security mathematically, and prove that the scheme is secure mathematically. At first glance, this mathematical approach to the analysis of cryptographic schemes, which has come to be known as *provable security*, may sound like it should be a straightforward task. It is not. For example, when defining security features, it is difficult to model all the potential threats against a cryptographic scheme with mathematics. We will never claim that provable security is a perfect vehicle to assess cryptographic security, but it is certainly a much better one than the informal approaches of the past.

The first attempt to prove the security of a public-key encryption scheme was by Rabin [31] in 1979, who described an encryption scheme for which recovering the message was as intractable as factoring an RSA modulus. Later, Goldwasser and Micali [19] described a scheme for which they could prove that extracting any information about the plaintext is as hard as deciding quadratic residuosity modulo composite numbers whose factorization is unknown. However, it was not until the early 1990s that researchers began to establish reliable and easy-to-use formal models for the security of an encryption scheme, and that the cryptographic community began to think about constructing practical and efficient provably-secure public-key encryption schemes. A brief history of provable security is provided in [13].

Provable security can be thought of as a tool for establishing the security of a cryptographic scheme. The basic idea is as follows:

1. Identify or establish an appropriate security model or, if necessary, define a new one.

2. Identify a hard mathematical problem to which the scheme to be assessed may relate.

3. The security proof starts by supposing that the scheme is not secure, i.e., that there exists a real-life adversary that can break the security of the scheme. Then construct an efficient algorithm that uses the adversary as

a subroutine to solve the hard mathematical problem. Since there should not exist any efficient algorithm that can solve the hard mathematical problem, by contradiction, the scheme is secure.

The term "security model" refers to a definition that states a security feature of a cryptographic scheme mathematically. A hard mathematical problem is often a computational problem such as the integer factorisation problem. A formal treatment is given in Chapter 2, where we provide security models for most cryptographic primitives and a list of some hard problems.

The provable security process is not straightforward at any stage. Defining a new security model requires care to make sure that the model indeed captures the required security features mathematically. Identifying the related hard mathematical problem is sometimes not easy, and it only makes sense if the identified problem is well-known to be hard or somehow related to a well-known to be hard mathematical problem. This is a crucial step since the entire security argument is based on the assumption that the underlying mathematical problem has no efficient algorithm that can solve it. Constructing the algorithm that solves the hard problem using a successful adversary is the key to establishing the proof, and is certainly not the easiest part.

1.2.1 Hash Functions and Random Oracles

A fairly controversial issue in provable security research is the use of the *random oracle model* [7]. The random oracle model assumes that hash functions used in cryptosystems are totally random functions and that the adversary has no information about their internal computations. The adversary may evaluate a random hash function by querying an oracle. This theoretical approach massively simplifies the analysis of cryptosystems and allows many cryptographic schemes to be proven secure that would otherwise be too complicated to be proven secure.

However, if a scheme is proven secure in the random oracle model, then the security argument only applies when the random oracle can be instantiated with an appropriate hash function, i.e., a hash function whose properties can justify the assumption. Moreover, Canetti *et al.* [10] demonstrated that it was possible to have a scheme that was provably secure in the random oracle model, yet insecure when the random oracle was replaced with any hash function. Canetti *et al.*'s trick is to contain a description of the hash function as part of an oracle query and the oracle is defined to return the private key as the response to this particular query. Despite the fact that the construction of such a scheme is artificial, it is a major problem.

Dent [12] looked into the problem and analysed the separation between the random oracle model and the *standard model*, where the standard model refers to not using the random oracle model. He suggested that if one can show that the situation described by Canetti *et al.* does not occur then a proof of security in the random oracle model should be sufficient. We take this as granted in this thesis, as all our schemes are "reasonable", i.e., there are no rules such as returning the private key if the ciphertext contains a description of the hash function.

As a methodology of provable security, the random oracle model has its limitations, but it has been widely adopted, and has made a significant contribution to theoretical cryptography. We believe that the random oracle model does provide us with a good understanding of certain cryptographic schemes and can be used to provide proven security at least to some extent.

1.3 Signcryption

Suppose that Alice would like to send messages to Bob. Our security goal throughout this thesis is to provide *confidentiality*, *integrity* and *data origin authentication* of messages between Alice and Bob.

One traditional approach for Alice to try to achieve these security guarantees is to:

1. sign the message using a digital signature scheme, then;
2. encrypt the signed message using a public-key encryption scheme.

This is called *sign-then-encrypt* (StE). Another traditional approach may include *encrypt-then-sign* (EtS), that is to:

1. encrypt the message using a public-key encryption scheme, then;
2. sign the encrypted message using a digital signature scheme.

Both of these approaches are a serial composition of two components: digital signatures and public-key encryption. They require the same amount of computational and communication overheads as the sum of the overheads for the digital signature scheme and for the public-key encryption scheme. This motivated researchers to find a more efficient solution, namely, a scheme that can allow two actions to be done more efficiently than just a serial composition.

Signcryption was introduced by Zheng in 1997 [37]. It is a shorter word for the expression "signature and encryption". As the name suggests, it is designed to gain computational efficiency as well as to provide a shorter output that represents both ciphertext and signature than the traditional approaches do. It is common practice to refer to the output as the *signcryption ciphertext*, or just *ciphertext* where there is no ambiguity.

A concrete signcryption scheme was also proposed by Zheng in [37]. It involves a single-step approach that achieves the same security objectives as StE or EtS, but requires less computational resources and is more bandwidth-efficient. A single word "*signcrypt*" is used to represent "sign and encrypt" to emphasise it is a single-step action. Correspondingly, the word "*unsigncrypt*" is used to represent "verify and decrypt". The scheme in [37] is a cunning combination of a digital signature scheme and a symmetric encryption scheme with shared randomness. The replacement of a public-key encryption scheme by a symmetric one and the reuse of randomness are critical to the efficiency improvements.

Since then, various extensions, refinements and adaptations of Zheng's techniques and the concept of signcryption have been made. For example, we now have signcryption schemes with public verifiability [18], signcryption schemes with non-repudiation [27, 33], signcryption schemes with anonymity [26], identity-based signcryption [11, 25], certificateless signcryption [5], deterministic signcryption [14], parallel signcryption [29], hybrid signcryption [9], and many others. A full range of signcryption schemes can be found in [15].

1.3.1 Security Models For Signcryption

There is another strand of research in signcryption that concentrates on the provable security of signcryption. This involves building computational security models to formalise various security guarantees so that one can mathematically prove security properties.

The first attempt to produce security models for signcryption [35] came surprisingly late, three years after the invention of signcryption. It only provided a security model that captures the *unforgeability* of a signcryption scheme, where "unforgeability" is conventionally used to refer to "integrity and data origin authentication" of a signcryption scheme (see Definition 2.3.11). A complete treatment on the security models of signcryption was independently provided by An *et al.* [2] and Baek *et. al.* [3] in 2002. Zheng's signcryption scheme was then formally proved secure in [3]. Both An *et al.*'s and Baek *et al.*'s security models capture the two main security objectives of a signcryption scheme: confidentiality and unforgeability. An *et al.* considered a two-user setting for the security models, where they assume that there are only two users of a signcryption scheme, and then they generalised this idea to a multi-user setting. Baek *et al.*, on the other hand, directly provided a multi-user setting for the security models, where the security of a signcryption scheme that has more than two users is defined. We adopt Baek *et al.*'s multi-user security model in this thesis.

As research in this direction continues, the security models are modified to suit various scenarios and more security models are built for special security features such as anonymity and ciphertext unlinkability. Based on these models, many formal security proofs have been produced for signcryption schemes. A full survey can be found in [15].

1.4 Outline of Thesis

In this section, we provide an outline of the thesis and emphasise our contributions.

1.4.1 The Forking Lemma

Some researchers look for generic techniques that can be adapted to prove that a large class of cryptographic schemes are secure, so that providing a security proof for such schemes becomes more convenient.

Pointcheval and Stern [30] introduced a generic proof technique called the *oracle replay attack*, and used the so-called *forking lemma* (see Theorem 3.2.3) to prove a large class of signature schemes secure.

The oracle replay attack can only be used in the random oracle model, since "oracle" refers to the random oracle. The idea is to replay a successful attack from an adversary against a signature scheme. The adversary is regarded as a probabilistic algorithm. A replay of a successful attack means to run the algorithm with the same random input again. However, during the replay, a different random oracle will be used, so that when the random oracle is queried with the same input, it is likely to get a different output. If the replay is also successful, then the forking lemma can provide a so-called forking algorithm that uses the adversary as a subroutine to solve a hard problem, such as the discrete logarithm problem, efficiently. We adapt this technique and use the forking lemma to prove that the *Zheng signature scheme* [38] is secure. A description of the Zheng signature scheme and the security proof are provided in Chapter 3.

Bellare and Neven [6] generalised Pointcheval and Stern's idea and provide the *general forking lemma* (see Theorem 4.7.4) for use in security proofs. We will use the general forking lemma when proving unforgeablity of some signcryption schemes.

1.4.2 Meta-ElGamal Signcryption Schemes

In this work, we will concentrate mainly on Zheng's signcryption scheme [37] and one of its variants, Gamage *et al.*'s signcryption scheme [18]. These two schemes can be thought of as applying similar generic transforms to a specific signature scheme called the *shortened Digital Signature Standard* (SDSS), introduced by Zheng in [37]. The SDSS signature scheme is one variant of the ElGamal signature scheme [17] which was identified by Horster *et al.* [23]. Horster *et al.* claimed to identify over 5,000 different variants of the ElGamal signature scheme and named these *meta-ElGamal signature schemes.*

According to Zheng [37], most of the meta-ElGamal signature schemes can be turned into signcryption schemes. Based on these ideas, we propose two generic transforms from meta-ElGamal signature schemes to signcryption schemes. This gives us a large class of signcryption schemes, which we call *meta-ElGamal signcryption schemes.* These signcryption schemes achieve the same security objectives and provide similar bandwidth and computational savings as Zheng's signcryption scheme or Gamage *et al.*'s signcryption scheme. We provide the full security proofs for each transform and a comparison between the two transforms in Chapter 4.

1.4.3 Outsider versus Insider

We consider users of signcryption schemes as senders and receivers. Since we are in the public-key setting, the sender and the receiver do not share the same secret key but, rather, each has their own private key. Thus, it makes sense to protect data not only from an outsider (third party) but also from an insider (a legal user of the system such as the sender or the receiver). This informs an additional security notion which we call *insider security*, while security against an outsider is referred to as *outsider security*. When we talk about the security features of a signcryption scheme, we add "outsider" or "insider" to indicate which adversary the signcryption scheme is resistant to, for example, *outsider confidentiality* or *insider unforgeability*. We also refer to security models as the *insider model* or the *outsider model* depending on against whom the security feature is defined. Formal definitions will be provided in Chapter 2.

It is tempting to think that insider security supersedes outsider security in terms of security measures. However, this is not exactly the case. In fact, not achieving insider confidentiality can sometimes be considered as a positive feature. For example, in some signcryption scheme with outsider confidentiality, it is possible for the sender to unsigncrypt messages using their private key [38]. This property, known as *past message recovery*, allows the *sender* to store ciphertexts and unsigncrypt them in the future when desired. On the other hand, achieving insider confidentiality can provide *forward secrecy*, i.e., provide security in the special circumstance where the sender's private key is compromised by an adversary who wants to unsigncrypt the ciphertexts signcrypted by this sender. In general, it tends to be easier to achieve outsider confidentiality, and many signcryption schemes are secure in the outsider model but not secure in the insider model. In the cases that both outsider security and insider security are sufficient, it is easier and more efficient to use outsider secure signcryption schemes.

Note that Zheng's signcryption scheme is proven to have outsider confidentiality and insider unforgeability, but cannot achieve insider confidentiality. This is also the case for our meta-ElGamal signcryption schemes. However, we will propose a modification of meta-ElGamal signcryption schemes so that the resulting signcryption schemes are secure against an insider. In particular, Zheng's signcryption scheme with this modification will have insider confidentiality and unforgeability, which is one of our contributions. As this extra security comes at the cost of only one more exponentiation, it is a reasonable trade-off in many application environments. Formal security proofs are all presented in Chapter 5.

1.4.4 Two-Key versus One-Key

A user of signcryption may wish to send messages to other users, as well as to receive messages from other users. In general, we assume that each user has two independent key pairs, one for sending and the other for receiving. In most signcryption schemes, the sender and receiver's key generation algorithms are identical, and a user may wish to use a single key pair for both sending and receiving. However, this single key pair generation setting opens up additional capabilities for attacks [15] and requires some modifications to the security models. Therefore, it is important to state whether single key pair generation is used or not when analyzing the security of a signcryption scheme.

We use the notions *one-key* or *two-key*, to indicate whether a signcryption scheme requires single key pair generation or two key pair generation, and therefore which security models are being used. If not stated explicitly, the two-key setting is implied. It should be noted that a two-key signcryption scheme with identical key generation algorithms which is secure in the two-key security model may not be secure in the one-key model. In particular, trivial "key concatenation" will result in an insecure one-key signcryption scheme. We will explain this clearly in Chapter 6, where we also propose a modification to trivial key concatenation and obtain a generic transform from a two-key signcryption scheme to a one-key signcryption scheme without any loss of security.

A one-key signcryption scheme allows users to use the same public/private key pair for both sending and receiving data. An interesting fact is that an insider secure one-key signcryption scheme results in a digital signature scheme and public-key encryption scheme pair, each of which can share the same public/private key pair securely.

It is conventional wisdom that keys used by cryptographic schemes should be different, indeed normally independent. However, Haber and Pinkas [22] initiated a formal study of security of key reuse and introduced the concept of a *combined public-key scheme*, where an encryption scheme and a signature scheme are combined. They considered various well-known cryptographic schemes and conditions under which their keys could be partially shared. Paterson *et al.* [28], on the other hand, took a different approach and proposed a general construction for a combined public-key scheme with joint security in the standard model. Under their construction, keys are completely shared among the public-key schemes.

It is worth pointing out here that our goal is *not* to find an efficient construction of a secure combined public-key scheme. Instead, we offer a generic transform from secure two-key signcryption to secure one-key signcryption to fill in the gap caused by the "failure" of trivial key concatenation. As a result, although our pair of public-key schemes can securely share identical keys, they are not efficient in general. More discussion will be provided in Chapter 6.

Chapter 2

Preliminary

In this chapter, we will explain some basic notation and some fundamental concepts in cryptography. We will also provide formal definitions for some cryptographic primitives and their corresponding security models.

2.1 Basic Notation and Concepts

2.1.1 Basic Mathematics

- Let a, b be any non-zero integers. We denote the *greatest common divisor* of a and b as $gcd(a, b)$. We write $a|b$ if a divides b.

- For any integer $N \geq 2$, we let $\mathbb{Z}_N^*$ denote the multiplicative group of $\mathbb{Z}_N$.

- Without ambiguity, we let $\{0, 1\}^*$ represent the set of all strings made of 0 or 1.

- Let E_1 and E_2 be any two events. Then $Pr[E_1]$ denotes the probability of the occurrence of E_1, and $Pr[E_1|E_2]$ denotes the conditional probability of E_1 happening given that E_2 happens.

2.1.2 Complexity Theory

When we analyze a cryptographic protocol or a computational problem, we often regard an entity involved in the process as a *probabilistic Turing machine*. A probabilistic Turing machine is a *Turing machine* [36] with an extra tape, called the *random tape*, which contains an infinite sequence of uniformly distributed independent random bits.

We say an algorithm is a *probabilistic polynomial-time* (PPT) algorithm, if it is probabilistic and its running time is polynomial in the size of the input.

When we say *PPT adversary*, it means that the adversary is regarded as a probabilistic Turing machine that has running time polynomial in the size of the input.

For more complexity theory, readers can refer to [36].

2.1.3 Algorithm and Assignment

We let $\leftarrow$ denote assignment and $\overset{\$}{\leftarrow}$ denote randomised assignment. Thus, if S is a finite set, then $y \overset{\$}{\leftarrow} S$ denotes the assignment to y of a uniformly random element of S.

If $\mathcal{A}$ is a deterministic algorithm then $y \leftarrow \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ run on x. Similarly, if $\mathcal{A}$ is a probabilistic algorithm then $y \overset{\$}{\leftarrow} \mathcal{A}(x)$ denotes the assignment to y of the output of $\mathcal{A}$ run on x and some new random bits.

Lastly, $y \leftarrow \mathcal{A}(x; \rho)$ denotes the assignment to y of the output of $\mathcal{A}$ run on x and a random tape ρ.

2.1.4 Security Parameter

In cryptography, a *security parameter* is a variable that measures the input size of a problem. The security parameter is usually expressed in unary representation, so that the efficiency of algorithms can be polynomial-time in the length of the input. We write 1^k to denote the unary representation of k, where

k is some positive integer. We use the letter k to denote security parameters throughout the thesis.

2.1.5 Negligible Function

A function $f\colon \mathbb{N} \to \mathbb{R}$ is *negligible* if for any polynomial p, there is an integer k_0 such that for all $k > k_0$, we have

$$f(k) < \frac{1}{|p(k)|}.$$

2.2 Intractability Assumptions

A problem is *tractable* if and only if there exists a probabilistic polynomial time algorithm for its solution. Formally, we make the following definition.

Definition 2.2.1. *Let $\mathcal{A}$ be an algorithm to solve a problem $\mathcal{P}$. We define the* advantage *of $\mathcal{A}$ against the problem $\mathcal{P}$ with input size k to be:*

$$Adv^{\mathcal{P}}_{\mathcal{A}}[k] = Pr[\mathcal{A} \text{ outputs a correct solution to a random instantiation of } \mathcal{P}].$$

We say the problem $\mathcal{P}$ is intractable *or* hard*, if every PPT $\mathcal{A}$ has negligible advantage.*

Hence, a problem is *intractable* if there does not exist any probabilistic polynomial time algorithm to solve it. It remains an open problem to actually prove that a certain problem is intractable. However, there is some evidence showing some problems are likely to be intractable. Based on this evidence, we often make assumptions that a certain computational problem is intractable. We now identify seven problems on which we make intractability assumptions in this thesis.

Definition 2.2.2 (Discrete Logarithm Problem in a Prime-Order Group)**.** *Let p be a k-bit prime. Let $\mathbb{G}$ be a cyclic group of order p with a random generator g. The* Discrete Logarithm *problem (DL) in $\mathbb{G}$ is to compute x given $g^x \in \mathbb{G}$ for randomly chosen $x \in \mathbb{Z}_p^*$.*

Definition 2.2.3 (Discrete Logarithm Problem in a Composite-Order Group)**.** *Let N be a k-bit composite number. Let $\mathbb{G}$ be a group of order N. Let $\mathbb{G}'$ be a subgroup of $\mathbb{G}$ with a generator g. The* Discrete Logarithm *problem (DL) in the composite-order group is to compute x given a random element $h \in \mathbb{G}'$ such that $g^x = h$.*

Definition 2.2.4 (Computational Diffie-Hellman Problem)**.** *Let p be a k-bit prime. Let $\mathbb{G}$ be a cyclic group of order p with a random generator g. The* Computational Diffie-Hellman *problem (CDH) is to compute $h = g^{ab}$ given $(g^a, g^b) \in \mathbb{G}^2$ for randomly chosen $a, b \in \mathbb{Z}_p^*$.*

Definition 2.2.5 (Decisional Diffie-Hellman Problem)**.** *Let p be a k-bit prime. Let $\mathbb{G}$ be a cyclic group of order p with a random generator g. The* Decisional Diffie-Hellman *problem (DDH) is to distinguish the distribution of* Diffie-Hellman tuples*, $D_{DH} := \{g^a, g^b, g^{ab} | a, b \xleftarrow{\$} \mathbb{Z}_p^*\}$, from that of random tuples, $D_{rand} := \{g^a, g^b, g^c | a, b, c \xleftarrow{\$} \mathbb{Z}_p^*\}$.*

Definition 2.2.6 (Integer Factorisation)**.** *Let p, q be two randomly chosen k-bit primes. Let N be the product of p and q. The* Integer Factorisation *problem is to compute p, q given N.*

The last two problems are introduced in [3]. They are slightly different from those above. Both of the two problems involve a *decisional Diffie-Hellman oracle*, which can be thought of as a black box that solves the decisional Diffie-Hellman problem.

Definition 2.2.7 (Gap Discrete Logarithm Problem)**.** *Let p be a k-bit prime. Let $\mathbb{G}$ be a cyclic group of order p with a random generator g. A* Decisional Diffie-Hellman *(DDH)* oracle*, $\mathcal{O}$, is defined to be an oracle with input $(g^x, g^y, g^z) \in \mathbb{G}^3$, and output 1 if $z = xy$ and 0 otherwise. The* Gap Discrete Logarithm *problem (GDL) is the DL problem in $\mathbb{G}$ with access to a DDH oracle $\mathcal{O}$.*

Definition 2.2.8 (Gap Diffie-Hellman Problem). *Let p be a k-bit prime. Let $\mathbb{G}$ be a cyclic group of order p with a random generator g. The* Gap Diffie-Hellman *problem (GDH) is the CDH problem in $\mathbb{G}$ with access to a DDH oracle $\mathcal{O}$.*

2.3 Cryptographic Primitives and Security Models

2.3.1 Symmetric Encryption Schemes

Definition 2.3.1. *A* symmetric encryption scheme *with security parameter k is defined by a pair of deterministic algorithms $(\mathcal{E}_{SE}, \mathcal{D}_{SE})$ as follows:*

- $\mathcal{E}_{SE}(K, m)$*: This is the encryption algorithm that takes as input a key $K \in \{0,1\}^\ell$ and a message $m \in \mathcal{M}$, and outputs a ciphertext $C \in \mathcal{C}$, where ℓ is an integer determined by k, $\mathcal{M}$ is some message space, and $\mathcal{C}$ is some ciphertext space.*

- $\mathcal{D}_{SE}(K, C)$*: This is the decryption algorithm that takes as input a key $K \in \{0,1\}^\ell$ and a ciphertext $C \in \mathcal{C}$, and outputs a message $m \in \mathcal{M}$ or $\perp$.*

It is required that $\mathcal{D}_{SE}(K, \mathcal{E}_{SE}(K, m)) = m$ for all $K \in \{0,1\}^\ell$ and $m \in \mathcal{M}$.

In this thesis, we also require that for any pair $(m, c) \in \mathcal{M} \times \mathcal{C}$, the probability that a randomly chosen $K \in \{0,1\}^\ell$ satisfies $\mathcal{E}_{SE}(K, m) = c$ is negligible in k.

It is natural to have several security models for a particular cryptographic primitive that capture different levels of security, i.e., weak security or strong security, so that users can sometimes trade off between security and efficiency. The strength of security normally depends on how powerful the adversary is assumed to be. We choose the security models that will be used later in the thesis to be presented here. As the security of our signcryption schemes depend on the security of a symmetric encryption scheme, we provide the security model that captures the required security notion, one-time indistinguishability, IND-OT, as follows:

Definition 2.3.2. *Let $SE = (\mathcal{E}_{SE}, \mathcal{D}_{SE})$ be a symmetric encryption scheme with security parameter k. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary against SE. We define the IND-OT security game* $\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{b}}(k)$*, where $b \in \{0,1\}$, as follows:*

```
$\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{b}}(k)$:
  $K \xleftarrow{\$} \{0,1\}^{\ell}$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1(k)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  Else $C^* \leftarrow \mathcal{E}_{SE}(K, m_b)$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  Output $b'$.
```

The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{\mathtt{OT}}(k) = |\Pr[\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{1}}(k) = 1] - \Pr[\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{0}}(k) = 1]|.$$

We say SE is IND-OT *secure if every PPT adversary has negligible advantage.*

A game such as $\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{b}}(k)$ is played between an adversary and a challenger, where the adversary is indicated by the subscript. The goal of the adversary is to break the security feature indicated by the superscript. As this is the first time of such presentation of a game appearing in the thesis, an interpretation of the game $\textsc{Expt}_{\mathcal{A}}^{\mathtt{OT}\text{-}\mathtt{b}}(k)$ is provided as follows:

1. The challenger randomly generates a key $K \in \{0,1\}^{\ell}$.

2. The adversary $\mathcal{A}_1$ is given the security parameter k, and generates two messages with equal length m_0 and m_1, together with some state information ω. The state information may include some information that $\mathcal{A}_1$ has collected or knowledge that $\mathcal{A}_1$ wishes to share with $\mathcal{A}_2$. m_0 and m_1 are then passed to the challenger, and ω is passed to $\mathcal{A}_2$.

3. On receiving the messages, the challenger checks if they have equal length. If not, the challenger sets C^* as $\perp$.

4. Otherwise, the challenger encrypts the message m_b using the key K, and obtains the challenge ciphertext C^*. The challenger passes C^* to the adversary $\mathcal{A}_2$.

5. The adversary $\mathcal{A}_2$ receives C^* and outputs a guess b', where b' indicates that the adversary thinks that the message $m_{b'}$ is the plaintext of C^*.

Note that b' is the output of the game $\textsc{Expt}_{\mathcal{A}}^{\texttt{OT-b}}(k)$.

2.3.2 Digital Signature Schemes

Definition 2.3.3. *A* digital signature scheme *is defined by a tuple of PPT algorithms* $(Setup_{DS}, \mathcal{G}_{DS}, \mathcal{S}_{DS}, \mathcal{V}_{DS})$ *as follows:*

- $Setup_{DS}(1^k)$*: This is the setup algorithm that takes as input a security parameter* 1^k*, and outputs some public parameters* PP*.*

- $\mathcal{G}_{DS}(PP)$*: This is the key generation algorithm that takes as input the public parameters* PP*, and outputs a public/private key pair* (pk, sk)*.*

- $\mathcal{S}_{DS}(sk, m)$*: This is the signing algorithm that takes as input the private key* sk *and a message* $m \in \mathcal{M}$*, and outputs a signature* σ*, where* $\mathcal{M}$ *is some message space.*

- $\mathcal{V}_{DS}(pk, m, \sigma)$*: This is the verification algorithm that takes as input the public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* σ*, and outputs either a valid symbol* $\top$ *or an invalid symbol* $\perp$*.*

It is required that for all $(pk, sk) \overset{\$}{\leftarrow} \mathcal{G}_{DS}(1^k)$ *and* $m \in \mathcal{M}$*, we have that*

$$\mathcal{V}_{DS}(pk, m, \mathcal{S}_{DS}(sk, m)) = \top$$

with probability 1*.*

Some signature schemes do not have the setup algorithm. In this case, the key generation algorithm takes as input the security parameter and outputs a key pair. As a convention in this work, we do not emphasise the difference between the two cases.

The security that we consider here for a signature scheme is UF-CMA security [20]:

Definition 2.3.4. *A signature scheme* $(Setup_{DS}, \mathcal{G}_{DS}, \mathcal{S}_{DS}, \mathcal{V}_{DS})$ *is* UF-CMA *secure if for any PPT adversary* $\mathcal{A}$ *the probability that* $\Pr[\textsc{Expt}^{UF}_{\mathcal{A}}(k) = 1]$ *is negligible as a function of* k*, where* $\textsc{Expt}^{UF}_{\mathcal{A}}(k)$ *is defined as follows:*

```
$\textsc{Expt}^{UF}_{\mathcal{A}}(k)$:
  $PP \xleftarrow{\$} Setup_{DS}(1^k)$
  $(pk, sk) \xleftarrow{\$} \mathcal{G}_{DS}(PP)$
  $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{S}_{DS}(sk,\cdot)}(pk)$
  Output 1 if
    (a) $\mathcal{V}_{DS}(pk, m^*, \sigma^*) = \top$
    (b) $\mathcal{A}$ never queried $\mathcal{S}_{DS}(sk, m^*)$
  Else output 0.
```

It is worth noting that there is another security notion for digital signatures, *strong unforgeability under chosen message attack* (sUF-CMA) [2]. Instead of restricting that $\mathcal{A}$ cannot query $\mathcal{S}_{DS}(sk, m^*)$, the sUF-CMA model only forbids the situations where $\mathcal{A}$ made a $\mathcal{S}_{DS}(sk, m^*)$ query *and* received (m^*, σ^*) as the respond. Unfortunately, we are only able to achieve UF-CMA security for the digital signature schemes in this work. However, we do not have any proof that these digital signature schemes cannot achieve sUF-CMA.

2.3.3 Public-Key Encryption Schemes

Definition 2.3.5. *A* public-key encryption scheme *is defined by a triple of PPT algorithms* $(\mathcal{G}_{PKE}, \mathcal{E}_{PKE}, \mathcal{D}_{PKE})$ *as follows:*

- $\mathcal{G}_{PKE}(1^k)$*: This is the key generation algorithm that takes as input a security parameter* 1^k*, and outputs a public/private key pair* (pk, sk)*.*

- $\mathcal{E}_{PKE}(pk, m)$: *This is the encryption algorithm that takes as input the public key* pk *and a message* $m \in \mathcal{M}$, *and outputs a ciphertext* $C \in \mathcal{C}$, *where* $\mathcal{M}$ *is some message space and* $\mathcal{C}$ *is some ciphertext space.*

- $\mathcal{D}_{PKE}(sk, C)$: *This is the decryption algorithm that takes as input the private key* sk *and a ciphertext* $C \in \mathcal{C}$, *and outputs a message* $m \in \mathcal{M}$ *or* $\perp$.

It is required that for all $(pk, sk) \xleftarrow{\$} \mathcal{G}_{PKE}(1^k)$ *and* $m \in \mathcal{M}$, *we have that*

$$\mathcal{D}_{PKE}(sk, \mathcal{E}_{PKE}(pk, m)) = m\,.$$

The security notion for public-key encryption schemes we choose to present here is IND-CCA2, which is defined as follows:

Definition 2.3.6. *Let* $(\mathcal{G}_{PKE}, \mathcal{E}_{PKE}, \mathcal{D}_{PKE})$ *be a public-key encryption scheme. Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a two-stage adversary against the confidentiality of the public-key encryption scheme. Let* k *be a security parameter.* $\text{Expt}_{\mathcal{A}}^{\textit{IND-b}}(k)$ *is then defined as follows:*

```
$\text{Expt}_{\mathcal{A}}^{\textit{IND-b}}(k)$:
  $(pk, sk) \xleftarrow{\$} \mathcal{G}_{PKE}(1^k)$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{D}_{PKE}(sk,\cdot)}(pk)$
  $C^* \xleftarrow{\$} \mathcal{E}_{PKE}(pk, m_b)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{D}_{PKE}(sk,\cdot)}(C^*, \omega)$
  Output $b'$,
```

where $\mathcal{A}_2$ *cannot make a decryption query for* C^*. *Then the adversary's advantage is defined as:*

$$Adv_{\mathcal{A}}^{\textit{IND}}(k) = \left|\Pr[\text{Expt}_{\mathcal{A}}^{\textit{IND-1}}(k) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\textit{IND-0}}(k) = 1]\right|\,.$$

We say the public-key encryption scheme is IND-CCA2 *secure if every PPT adversary* $\mathcal{A}$ *has negligible advantage in* k.

2.3.4 Signcryption Schemes

Definition 2.3.7. *A* signcryption scheme *is a tuple of PPT algorithms* $(Setup, KG_S, KG_R, SC, USC)$*, where*

- *the setup algorithm produces public parameters* $PP \xleftarrow{\$} Setup(1^k)$ *for the security level* k*;*
- *the sender key-generation algorithm produces a sender key pair* $(pk_S, sk_S) \xleftarrow{\$} KG_S(PP)$*;*
- *the receiver key-generation algorithm produces a receiver key pair* $(pk_R, sk_R) \xleftarrow{\$} KG_R(PP)$*;*
- *the signcryption algorithm takes as input a message* m *from some message space* $\mathcal{M}$*, the sender private key* sk_S *and the receiver public key* pk_R*, and outputs a signcryption ciphertext* $C \xleftarrow{\$} SC(PP, sk_S, pk_S, pk_R, m)$ *in a ciphertext space* $\mathcal{C}$*; and*
- *the unsigncryption algorithm takes as input a ciphertext* $C \in \mathcal{C}$*, the sender public key* pk_S *and the receiver private key* sk_R*, and outputs either a message* $m \leftarrow USC(PP, pk_S, sk_R, pk_R, C)$ *or the error symbol* $\perp$*.*

We will generally assume that the public parameters PP *are an implicit input to all algorithms, rather than explicitly writing their input. For correctness, we require that for all public parameters* PP *and key pairs* $(pk_S, sk_S) \xleftarrow{\$} KG_S(1^k)$ *and* $(pk_R, sk_R) \xleftarrow{\$} KG_R(1^k)$*, and for all* $m \in \mathcal{M}$*, we have that*

$$USC(pk_S, sk_R, pk_R, SC(sk_S, pk_S.pk_R, m)) = m$$

with probability 1*.*

Signcryption can offer two main security features: confidentiality and unforgeability. We define two types of security models depending on whether the adversary is an insider or outsider.

Definition 2.3.8 (Outsider IND-CCA2). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT adversary against the confidentiality of a signcryption scheme with security parameter k. Then the* outsider confidentiality *of the signcryption scheme is captured via the security game* $\textsc{Expt}_{\mathcal{A}}^{\texttt{out-IND-b}}(k)$ *defined as follows:*

```
$\textsc{Expt}_{\mathcal{A}}^{\texttt{out-IND-b}}(k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(pk_S^*, sk_S^*) \xleftarrow{\$} KG_S(1^k)$
  $(pk_R^*, sk_R^*) \xleftarrow{\$} KG_R(1^k)$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_U}(pk_S^*, pk_R^*)$
  $C^* \xleftarrow{\$} SC(sk_S^*, pk_R^*, m_b)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_U}(C^*, \omega)$
  Output $b'$ ,
```

where the signcryption oracle $\mathcal{O}_S$ and the unsigncryption oracle $\mathcal{O}_U$ are defined as

$$\mathcal{O}_S(pk_R, m) = SC(sk_S^*, pk_R, m) \quad \textit{and} \quad \mathcal{O}_U(pk_S, C) = USC(pk_S, sk_R^*, C) \ ,$$

with the condition that $\mathcal{A}_2$ cannot query $\mathcal{O}_U(pk_S^*, C^*)$. The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{\texttt{out-IND}}(k) = \left|\Pr[\textsc{Expt}_{\mathcal{A}}^{\texttt{out-IND-1}}(k) = 1] - \Pr[\textsc{Expt}_{\mathcal{A}}^{\texttt{out-IND-0}}(k) = 1]\right| \ .$$

The signcryption scheme is said to be outsider IND-CCA2 *secure if $Adv_{\mathcal{A}}^{\texttt{out-IND}}(k)$ is negligible in k for every PPT adversary $\mathcal{A}$.*

It is worth noting that the adversary has the choices over the public keys pk_R and pk_S when making the signcryption query and the unsigncryption query respectively. This makes the security model as a multi-user model [3]

Definition 2.3.9 (Insider IND-CCA2). *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT two-stage adversary against the confidentiality of a signcryption scheme with security parameter k. Then the* insider confidentiality *of the signcryption scheme is captured via the security game* $\textsc{Expt}_{\mathcal{A}}^{\texttt{in-IND-b}}(k)$ *which is defined as follows:*

```
$\textsc{Expt}_{\mathcal{A}}^{\texttt{in-IND-b}}(k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(pk_R^*, sk_R^*) \xleftarrow{\$} KG_R(1^k)$
  $(m_0, m_1, sk_S^*, pk_S^*, \omega) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_U}(pk_R^*)$
  $C^* \xleftarrow{\$} SC(sk_S^*, pk_R^*, m_b)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}_U}(C^*, \omega)$
  Output $b'$ ,
```

where the unsigncryption oracle $\mathcal{O}_U$ *is defined as*

$$\mathcal{O}_U(pk_S, C) = USC(pk_S, sk_R^*, C) \ ,$$

with the condition that $\mathcal{A}_2$ *cannot query* $\mathcal{O}_U(pk_S^*, C^*)$. *The adversary's advantage is defined to be*

$$Adv_{\mathcal{A}}^{\texttt{in-IND}}(k) = |\Pr[\textsc{Expt}_{\mathcal{A}}^{\texttt{in-IND-1}}(k) = 1] - \Pr[\textsc{Expt}_{\mathcal{A}}^{\texttt{in-IND-0}}(k) = 1]|.$$

The signcryption scheme is said to be insider IND-CCA2 secure *if* $Adv_{\mathcal{A}}^{\texttt{in-IND}}(k)$ *is negligible in* k *for every PPT adversary* $\mathcal{A}$.

It is worth noting that $\mathcal{A}_2$ *can* query $\mathcal{O}_U(pk_S, C^*)$ for any $pk_S \neq pk_S^*$. The adversary has no access to a signcryption oracle, as the adversary is assumed to be an insider who has the knowledge of sender's secret key, and the adversary is attacking the targeted receiving key pair.

Definition 2.3.10 (Outsider UF-CMA). *Let* $\mathcal{A}$ *be a PPT adversary against the integrity of a signcryption scheme with security parameter* k. *Then the* outsider unforgeability *of the signcryption scheme is captured via the security game* $\textsc{Expt}_{\mathcal{A}}^{\texttt{out-UF}}(k)$ *which is defined as follows:*

```
$\textsc{Expt}_{\mathcal{A}}^{\texttt{out-UF}}(k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(pk_S^*, sk_S^*) \xleftarrow{\$} KG_S(1^k)$
  $(pk_R^*, sk_R^*) \xleftarrow{\$} KG_R(1^k)$
  $C^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_U}(pk_S^*, pk_R^*)$
  Output 1 if
    (a) $m^* \neq \perp$ for $m^* \leftarrow USC(pk_S^*, sk_R^*, C^*)$
    (b) $\mathcal{A}$ never queried $\mathcal{O}_S(pk_R^*, m^*)$
  Else output 0,
```

where the signcryption oracle $\mathcal{O}_S$ and the unsigncryption oracle $\mathcal{O}_U$ are defined as

$$\mathcal{O}_S(pk_R, m) = SC(sk_S^*, pk_R, m) \quad \text{and} \quad \mathcal{O}_U(pk_S, C) = USC(pk_S, sk_R^*, C) \ .$$

The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{out\text{-}UF}(k) = \Pr[\text{Expt}_{\mathcal{A}}^{out\text{-}UF}(k) = 1].$$

The signcryption scheme is said to be outsider UF-CMA *secure if* $Adv_{\mathcal{A}}^{out\text{-}UF}(k)$ *is negligible for every PPT adversary* $\mathcal{A}$.

Definition 2.3.11 (Insider UF-CMA)**.** *Let $\mathcal{A}$ be a PPT adversary against the integrity of a signcryption scheme with security parameter k. Then the insider unforgeability of the signcryption scheme is captured via the security game* $\text{Expt}_{\mathcal{A}}^{in\text{-}UF}(k)$ *which is defined as follows:*

```
$\text{Expt}_{\mathcal{A}}^{in\text{-}UF}(k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(pk_S^*, sk_S^*) \xleftarrow{\$} KG_S(1^k)$
  $(pk_R^*, sk_R^*, C^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_S}(pk_S^*)$
  Output 1 if
    (a) $m^* \neq \perp$ for $m^* \leftarrow USC(pk_S^*, sk_R^*, C^*)$
    (b) $\mathcal{A}$ never queried $\mathcal{O}_S(pk_R^*, m^*)$
  Else output 0,
```

where the signcryption oracle $\mathcal{O}_S$ is defined as

$$\mathcal{O}_S(pk_R, m) = SC(sk_S^*, pk_R, m) \ .$$

The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{in\text{-}UF}(k) = \Pr[\text{Expt}_{\mathcal{A}}^{in\text{-}UF}(k) = 1].$$

The signcryption scheme is said to be insider UF-CMA *secure if* $Adv_{\mathcal{A}}^{in\text{-}UF}(k)$ *is negligible for every PPT adversary* $\mathcal{A}$.

So far, our definition of signcryption involves two independent key generation algorithms, one for sending and the other for receiving. The users of signcryption then can be regarded as senders or receivers. It is likely that a user is a sender as well as a receiver. In this case, the user will posses two independent pairs of keys. However, it may be possible to use only one key generation algorithm and each user is given a single key pair for both sending and receiving. This one key generation setting opens additional capabilities for attack. Therefore we need to make some modifications to the security models. The formal definitions for one-key signcryption and its insider security are provided as follows:

Definition 2.3.12. *A* one-key signcryption *scheme is a tuple of PPT algorithms $(Setup, KG, SC, USC)$, where*

- *the setup algorithm produces public parameters $PP \xleftarrow{\$} Setup(1^k)$ for the security level k;*
- *the key-generation algorithm produces a key pair $(pk, sk) \xleftarrow{\$} KG(PP)$ for both sending and receiving;*
- *the signcryption algorithm takes as input a message m from some message space $\mathcal{M}$, a User's private key sk^A and another user's public key pk^B, and outputs a signcryption ciphertext $C \xleftarrow{\$} SC(PP, sk^A, pk^B, m)$ in a ciphertext space $\mathcal{C}$; and*
- *the unsigncryption algorithm takes as input a ciphertext $C \in \mathcal{C}$, a user's public key pk^A and another user's private key sk^B, and outputs either a message $m \leftarrow USC(PP, pk^A, sk^B, C)$ or the error symbol $\perp$.*

We will generally assume that the public parameters PP are an implicit input to all algorithms, rather than explicitly writing their input. For correctness, we require that for all public parameters PP and key pairs $(pk^A, sk^A) \xleftarrow{\$} KG(1^k)$

and $(pk^B, sk^B) \overset{\$}{\leftarrow} KG(1^k)$*, we have that* $USC(pk^A, sk^B, SC(sk^A, pk^B, m)) = m$ *with probability* 1.

We will be presenting the definitions of one-key insider security next (instead of one-key outsider security), as we are only interested in one-key signcryption schemes that are insider secure in this work.

Definition 2.3.13 (One-Key Insider IND-CCA2)**.** *Let* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *be a PPT two-stage adversary against the confidentiality of a one-key signcryption scheme with security parameter* k*. Then the* one-key insider confidentiality *of the one-key signcryption scheme is captured via the security game* $\text{Expt}_{\mathcal{A}}^{\texttt{One-in-IND-b}}(k)$ *which is defined as follows:*

```
$\text{Expt}_{\mathcal{A}}^{\texttt{One-in-IND-b}}(k)$:
  $PP \overset{\$}{\leftarrow} Setup(1^k)$
  $(pk^U, sk^U) \overset{\$}{\leftarrow} KG(1^k)$
  $(m_0, m_1, sk^*, pk^*, \omega) \overset{\$}{\leftarrow} \mathcal{A}_1^{\mathcal{O}_S, \mathcal{O}_U}(pk^U)$
  $C^* \overset{\$}{\leftarrow} SC(sk^*, pk^U, m_b)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \overset{\$}{\leftarrow} \mathcal{A}_2^{\mathcal{O}_S, \mathcal{O}_U}(C^*, \omega)$
  Output $b'$ ,
```

where the signcryption oracle $\mathcal{O}_S$ *and the unsigncryption oracle* $\mathcal{O}_U$ *are defined as*

$$\mathcal{O}_S(pk, m) = SC(sk^U, pk, m) \quad \text{and} \quad \mathcal{O}_U(pk, C) = USC(pk, sk^U, C)$$

with the condition that $\mathcal{A}_2$ *cannot query* $\mathcal{O}_U(pk^*, C^*)$*. The adversary's advantage is defined to be*

$$Adv_{\mathcal{A}}^{\texttt{One-in-IND}}(k) = |\Pr[\text{Expt}_{\mathcal{A}}^{\texttt{One-in-IND-1}}(k) = 1] - \Pr[\text{Expt}_{\mathcal{A}}^{\texttt{One-in-IND-0}}(k) = 1]|.$$

The one-key signcryption scheme is said to be one-key insider IND-CCA2 secure *if* $Adv_{\mathcal{A}}^{\texttt{One-in-IND}}(k)$ *is negligible in* k *for every PPT adversary* $\mathcal{A}$*.*

Note that the adversary is given two oracles, $\mathcal{O}_S$ and $\mathcal{O}_U$, which is different from the two-key insider IND-CCA2 model. This is because the targeted key pair has two functionalities, sending and receiving. The adversary may take advantage of this additional feature.

Definition 2.3.14 (One-Key Insider UF-CMA)**.** *Let $\mathcal{A}$ be a PPT adversary against the integrity of a one-key signcryption scheme with security parameter k. Then the* one-key insider unforgeability *of the one-key signcryption scheme is captured via the security game* $\text{Expt}_{\mathcal{A}}^{\texttt{One-in-UF}}(k)$ *which is defined as follows:*

```
$\text{Expt}_{\mathcal{A}}^{\texttt{One-in-UF}}(k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(pk^U, sk^U) \xleftarrow{\$} KG(1^k)$
  $(pk^*, sk^*, C^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_S, \mathcal{O}_U}(pk^U)$
  Output 1 if
    (a) $m^* \neq \perp$ for $m^* \leftarrow USC(pk^U, sk^*, C^*)$
    (b) $\mathcal{A}$ never queried $\mathcal{O}_S(pk^*, m^*)$
  Else output 0,
```

where the signcryption oracle $\mathcal{O}_S$ and the unsigncryption oracle $\mathcal{O}_U$ are defined as

$$\mathcal{O}_S(pk, m) = SC(sk^U, pk, m) \quad \textit{and} \quad \mathcal{O}_U(pk, C) = USC(pk, sk^U, C) \ .$$

The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{\texttt{One-in-UF}}(k) = \Pr[\text{Expt}_{\mathcal{A}}^{\texttt{One-in-UF}}(k) = 1].$$

The one-key signcryption scheme is said to be one-key insider UF-CMA *secure if $Adv_{\mathcal{A}}^{\texttt{One-in-UF}}(k)$ is negligible for every PPT adversary $\mathcal{A}$.*

2.4 Summary

In this chapter, we have provided formal definitions of some common cryptographic primitives as well as signcryption schemes. We have also formally defined appropriate security models. In the rest of the thesis, we will rely on these formal definitions to produce security proofs.

Chapter 3

A Security Proof Using The Forking Lemma

To extend the computational underpinnings of signcryption schemes to problems related to integer factorization, Steinfeld and Zheng [35] described a scheme which is provably unforgeable assuming the hardness of factoring. Later, Zheng [38] proposed an extension of this scheme, together with an identification scheme and a signature scheme, using high order residues modulo an RSA composite. However, a formal analysis of the security of these schemes was not provided. In this chapter, we provide a formal security proof for the signature scheme.

3.1 The Zheng Signature Scheme

The Zheng signature scheme is based on the Schnorr signature scheme which was proposed by C.P. Schnorr in 1989 [32]. The Schnorr signature scheme was originally designed for smart cards. It is very efficient and produces short signatures. The Schnorr signature scheme has been proven to be UF-CMA secure in the random oracle model assuming the hardness of the discrete logarithm problem in a prime-order group [30]. The Zheng signature scheme is similar to the Schnorr signature scheme. The difference is subtle, but fundamental. As a result of the difference, one cannot apply the security proof for the Schnorr signature scheme to the Zheng signature scheme directly. To provide a clear comparison, we will describe both signature schemes in this section.

Definition 3.1.1 (The Schnorr signature scheme)**.** *Let k be a security parameter. The* Schnorr signature scheme *is a tuple of algorithms* $(Setup, \mathcal{G}, \mathcal{S}, \mathcal{V})$, *where*

```
Setup(1^k):
  Generate a group $\mathbb{G}$ of order $q$
    where $q$ is a $k$-bit prime
  Choose a random generator $g \in \mathbb{G}$
  Choose a hash function
    $H : \{0,1\}^* \to \mathbb{Z}_q$
  $PP \leftarrow (g, q, H)$
  Output $PP$

$\mathcal{G}(PP)$:
  Parse $PP$ as $(g, q, H)$
  $x \xleftarrow{\$} \mathbb{Z}_q^*$
  $y \xleftarrow{\$} g^{-x}$
  $sk \leftarrow x$
  $pk \leftarrow y$
  Output $(sk, pk)$

$\mathcal{S}(sk, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $b \leftarrow H(m, w)$
  $s \leftarrow t + bx \mod q$
  $\sigma \leftarrow (b, s)$
  Output $\sigma$

$\mathcal{V}(pk, m, \sigma)$:
  Parse $\sigma$ as $(b, s)$
  $w' \leftarrow y^b g^s$
  If $H(m, w') = b$
    output $\top$
  Else
    output $\bot$
```

The Schnorr signature scheme works within a prime-order group as described above. The exponentiation stands for repeated applications of the group operation, and the addition and the multiplication are done modulo q.

Instead of working within a prime-order group, the Zheng signature scheme works within a composite-order group. Instead of working with a group generator, the Zheng signature scheme works with some special element in the composite-order group. To make the Zheng signature scheme and its security argument work, the composite and the element should be carefully chosen according to the following definition [38]:

Definition 3.1.2. *Let k be a security parameter. We say that three integers (r, n, h) are a* good triplet *if they fulfill the following requirements:*

1. r is a prime whose size (length in binary representation) is at least k bits;

2. $n = pq$ is an RSA modulus of at least $3k$ bits, satisfying $gcd(r, p-1) = r$ and $gcd(r, q-1) = 1$;

3. h is an r-th nonresidue modulo n, or equivalently, $h^{(p-1)/r} \neq 1 \bmod p$.

Note that we did not specify what the size of the security parameter k should be in the above definition. In [38], Zheng suggested that k should be at least 120 bits. The security argument from [30] which we are going to apply may require larger k. We will discuss the choice of k in detail at the end of this chapter.

The Zheng signature scheme can now be defined as follows [38]:

Definition 3.1.3 (The Zheng signature scheme)**.** *Let k be a security parameter. Ler $\mathcal{M}$ be some message space. The* Zheng signature scheme *is a tuple of algorithms $(Setup, \mathcal{G}, \mathcal{S}, \mathcal{V})$, where:*

```
$Setup(1^k)$:
  Generate a good triplet $(r, n, h)$
  Choose $\ell \in \mathbb{Z}$ such that $\ell > 2k$
  Choose a hash function
    $H : \mathcal{M} \times \mathbb{Z}_n \to \mathbb{Z}_{2^k}$
  $PP \leftarrow (n, h, \ell, H)$

$\mathcal{S}(sk, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_{2^{1.75\ell}}$
  $w \leftarrow h^t \mod n$
  $b \leftarrow H(m, w)$
  $s \leftarrow t + bx$
  $\sigma \leftarrow (b, s)$
  Output $\sigma$

$\mathcal{G}(PP)$:
  Parse $PP$ as $(n, h, \ell, H)$
  $x \xleftarrow{\$} \mathbb{Z}_{2^\ell}$
  $y \leftarrow h^{-x} \mod n$
  $sk \leftarrow x$
  $pk \leftarrow y$
  Output $(sk, pk)$

$\mathcal{V}(pk, m, \sigma)$:
  Parse $\sigma$ as $(b, s)$
  $w' \leftarrow y^b h^s \mod n$
  If $H(m, w') = b$
    output $\top$
  Else
    output $\bot$
```

There are three points to be addressed.

- It is worth noting that r is not used in the signature scheme after the good triplet is generated. In [38], Zheng suggested to pick the private key x in $\mathbb{Z}_{2^\ell}$ instead of $\mathbb{Z}_{2^r}$, where $\ell \geq |r| + 40$ and $|r|$ is the number of bits in the binary representation of r. We require ℓ to be at least $2k$ here, and we will explain the reason in the security proof.
- The Schnorr signature scheme works within $\mathbb{G}$, where $\mathbb{G}$ has a clear group structure as q and g are both publicly known. The Zheng signature scheme works within $\mathbb{Z}_n^*$ whose order is unknown as the factorization of n is kept secret. The element h also has unknown order in $\mathbb{Z}_n^*$. As a result, the computation of s, $s \leftarrow t + bx$, in the signing algorithm, can only be done as natural numbers.
- As the security proof for the Schnorr signature scheme uses the fact that $\mathbb{G}$ has prime order, we cannot apply the proof technique directly to the Zheng signature scheme, despite their apparent similarity.

3.2 The Security Proof

To formally analyze the security of the Zheng signature scheme, we need to first define what it means to be *secure*. In this case, we refer to Definition 2.3.4 and aim to prove that the Zheng signature scheme is UF-CMA secure. We adapt the definition in the context of the Zheng signature scheme here.

More precisely, we say that the Zheng signature scheme is UF-CMA secure if for any PPT algorithm $\mathcal{A}$ the probability that $\textsc{Expt}^{\texttt{UF}}_{\mathcal{A}}(k) = 1$ is negligible as a function of k, where $\textsc{Expt}^{\texttt{UF}}_{\mathcal{A}}(k)$ is defined as follows:

```
$\textsc{Expt}^{\mathtt{UF}}_{\mathcal{A}}(k)$:
  $PP \leftarrow Setup(1^k)$
  $(pk, sk) \leftarrow \mathcal{G}(PP)$
  $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{S}(sk,\cdot)}(pk)$
  Output 1 if
    (a) $\mathcal{V}(pk, m^*, \sigma^*) = acc$
    (b) $\mathcal{A}$ never queried $\mathcal{S}(sk, m^*)$
  Else output 0
```

The next step is to determine precisely what the underlying hard problem is. Our security proof shows that the underlying problem is the discrete logarithm problem in a composite-order group. In other words, we will prove that any efficient forgery of the Zheng signature scheme will lead to an efficient algorithm to solve the discrete logarithm problem in a composite-order group. It is worth noting that there is an efficient reduction from the composite discrete logarithm problem to the integer factorization problem [21].

Before starting the proof, we now briefly discuss the results in [30] and explain why we can apply those to the Zheng signature scheme.

3.2.1 The Oracle Replay Attack

In [30], Pointcheval and Stern introduced a generic reduction technique in order to provide security arguments for the signature schemes that they proposed. They named this technique the *oracle replay attack*.

To explain, suppose there is an attacker who has successfully forged a signature in the random oracle model. We can replay the attack with exactly the same parameters and randomness except that a different random oracle is used to interact with the attacker. We hope that the attacker will successfully forge another signature that is suitably related to the previous one. We can then extract the solution of a difficult algorithmic problem (e.g., discrete logarithm) from the ability to forge such signatures.

During this process, a major problem is to simulate properly the interactions that the attacker should have with other entities, in particular, the signer. We need to simulate the signer without the signer's private key and yet the attacker cannot distinguish between the signer and the simulator.

Formally, we model any participant by a probabilistic polynomial time Turing machine, and the communication tapes between the attacker and the simulator, and between the attacker and the signer, will have to follow indistinguishable distributions, where formal definitions of indistinguishability is taken from [30] as follows.

Definition 3.2.1. *Let δ^0, δ^1 be two distributions of probability. A* distinguisher *$\mathcal{D}$ is a probabilistic polynomial time Turing machine, with random tape ω, which, on input ρ, answers 0 or 1. The advantage of $\mathcal{D}$ with respect to two distributions δ^0 and δ^1 is defined as*

$$Adv(\mathcal{D}, \delta^0, \delta^1) = \frac{1}{2} \times |\mathbb{E}_{\rho \in \delta^0}[\mathcal{D}(\omega, \rho)] - \mathbb{E}_{\rho \in \delta^1}[\mathcal{D}(\omega, \rho)]| \, .$$

We say two distributions δ^0 and δ^1 are polynomially indistinguishable *if there does not exist any distinguisher $\mathcal{D}$ with a non-negligible advantage. We say two distributions δ^0 and δ^1 are* statistically indistinguishable *if*

$$\sum_y |Pr_{x \in \delta^0}[x = y] - Pr_{x \in \delta^1}[x = y]|$$

is negligible.

Note that if two distributions are statistically indistinguishable, they are polynomially indistinguishable [30].

3.2.2 The Forking Lemma

In [30], Pointcheval and Stern also proved a fundamental result for proving the security of digital signatures known as the *forking lemma*. We will apply the forking lemma directly to the Zheng signature scheme in order to prove its security. We skip the proof of the forking lemma here. Readers who are interested may refer to [30] for a detailed proof. However, we will verify that the forking lemma is indeed applicable to the Zheng signature scheme.

Pointcheval and Stern introduced the concept of *generic digital signature schemes*, which require the following conditions to be met:

1. Given an input message m, the signing algorithm will produce a triple of the form (w, b, s), where w randomly takes its values in a large set, b is the hash value of (m, w), and s depends on w and b.

2. The probability that w takes any particular value is no greater than 2^{1-k}, where k is the security parameter.

We now show that the Zheng signature scheme satisfies all the conditions and hence is a generic digital signature scheme.

1. For the first condition, we have $w = h^t \mod n$ for some random $t \in \mathbb{Z}_{2^{1.75\ell}}$, where 1.75ℓ is at least $3.5k$. Hence, $\mathbb{Z}_{2^{1.75\ell}}$ is a much larger set than the subgroup of $\mathbb{Z}_n^*$ generated by h. This implies that any particular value of w appears equally likely among the subgroup of $\mathbb{Z}_n^*$ generated by h.

2. Our next task is to prove that no w can appear with probability greater that 2^{1-k}. In fact, we will show that the order of h in $\mathbb{Z}_n^*$ is greater than or equal to r in Lemma 3.2.2. Hence, the subgroup of $\mathbb{Z}_n^*$ generated by h is of size at least r, i.e., at least 2^k. As w appears as a random element of the subgroup, the probability of each possible w is no greater than 2^{-k}, hence, less than 2^{1-k}. That is, the third condition met.

Lemma 3.2.2. *Let (r, n, h) be a good triplet with $n = pq$. Then the order of h in $\mathbb{Z}_n^*$ is greater than or equal to r.*

Proof: Let o_n denote the order of h in $\mathbb{Z}_n^*$ and o_p denote the order of h in $\mathbb{Z}_p^*$. It is clear that $o_n \geq o_p$. Now, we have $h^{o_p} \equiv 1 \mod p$ implies $o_p | p-1$. As r is a prime, we have either $(o_p, r) = 1$ or $(o_p, r) = r$. If $(o_p, r) = 1$, and as $o_p | p - 1$, we have $o_p | \frac{p-1}{r}$. This implies $h^{\frac{p-1}{r}} \equiv 1 \mod p$, a contradiction. If

$(o_p, r) = r$, then $o_p \geq r$. Hence, we have $o_n \geq o_p \geq r$. $\square$

We have thus verified that the Zheng signature scheme is a generic digital signature scheme as defined in [30]. Before we state the forking lemma, we note that Pointcheval and Stern [30] assumed that the hash function H takes values in $\mathbb{Z}_{2^k}$. This is indeed the case for the Zheng signature scheme.

Theorem 3.2.3 (The Forking Lemma [30]). *Let $(Setup, \mathcal{G}, \mathcal{S}, \mathcal{V})$ be a generic signature scheme with security parameter k. Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. We denote respectively by Q and R the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of queries that $\mathcal{A}$ can ask to the signer. Assume that, within a time bound T, $\mathcal{A}$ produces, with probability $\epsilon \geq 10(R+1)(R+Q)/2^k$, a valid signature (m, w, b, s). If the triples (w, b, s) can be simulated without knowing the private key with an indistinguishable distribution probability, then a replay of the attacker $\mathcal{A}$, where interactions with the signer are simulated, outputs two valid signatures (m, w, b, s) and (m, w, b', s') such that $b \neq b'$, within time $T' \leq 23QT/\epsilon$ and with probability $\epsilon' \geq 1/9$. Moreover, there is a machine which has control over the machine obtained from $\mathcal{A}$ replacing interaction with the signer by simulation and produces two valid signatures (m, y, b, s) and (m, y, b', s') such that $b \neq b'$ in expected time $T' \leq 120686QT/\epsilon$.*

3.2.3 The Proof

Theorem 3.2.4. *Let $\mathcal{A}$ be an adversary against UF-CMA security of the Zheng signature scheme with advantage ϵ and a time bound T. We denote respectively by Q and R the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of queries that $\mathcal{A}$ can ask to the signing oracle. Assume that $\epsilon \geq 10(R+1)(R+Q)/n$. Then the discrete logarithm in a subgroup of a composite-order group can be solved within expected time less than $120686QT/\epsilon$.*

Proof: We need to prove that the triples (w, b, s) can be simulated without knowing the private key and that they have an indistinguishable distribution from the attacker's point of view.

As we mentioned earlier, under the random oracle model, we assume the hash function $\mathcal{H}$ in the scheme is a truly random function. So from the attacker's point of view, the triples (w, b, s) have the following distribution:

$$\delta = \left\{ (w, b, s) \left| \begin{array}{l} t \in \mathbb{Z}_{2^{1.75\ell}} \\ b \in_R \mathbb{Z}_{2^k} \\ w = h^t \mod n \\ s = t + bx \end{array} \right. \right\}.$$

For simulation, we introduce a fake private key $x' \in \mathbb{Z}_\ell$ which is randomly chosen. We fix x', and consider the following distribution:

$$\delta' = \left\{ (w, b, s) \left| \begin{array}{l} t' \in_R \mathbb{Z}_{2^{1.75\ell}} \\ b \in_R \mathbb{Z}_{2^k} \\ s = t' + bx' \\ w = h^s y^b \mod n \end{array} \right. \right\}.$$

Note that the private key is not required to produce a valid triple (w, b, s). Now, we are going to prove that δ and δ' are statistically indistinguishable. Let (W, B, S) be the random variables with distribution δ and (W', B', S') be the random variables with distribution δ'. Let T, T' be two random variables uniformly distributed over $\mathbb{Z}_{2^{1.75\ell}}$. We observe that the random variables T and B determine W and S, and similarly T' and B' determine W' and S'. By the definition of statistical indistinguishability, we want to show that:

$$\sum_{(w,b,s)\in\delta\cup\delta'} |Pr[(W, B, S) = (w, b, s)] - Pr[(W', B', S') = (w, b, s)]| \qquad (*)$$

is negligible in k.

$$\begin{aligned}
&Pr[(W, B, S) = (w, b, s)] \\
&= Pr[W = w, B = b, S = s] \\
&= Pr[W = w|B = b, S = s]Pr[B = b, S = s] \\
&= Pr[W = w|B = b, S = s]Pr[S = s|B = b]Pr[B = b] \\
&= Pr[W = w|B = b, S = s]Pr[T = s - bx|B = b]Pr[B = b] \\
&= Pr[W = w|B = b, S = s]Pr[T = s - bx]Pr[B = b].
\end{aligned}$$

The second and the third equalities are conditional probability formulae. The fourth equality is true since $s = t + bx$. The last is true because the choice of T is independent of B.

Similarly, we have

$$\begin{aligned}
&Pr[(W', B', S') = (w, b, s)] \\
&= Pr[W' = w, B' = b, S' = s] \\
&= Pr[W' = w|B' = b, S' = s]Pr[B' = b, S' = s] \\
&= Pr[W' = w|B' = b, S' = s]Pr[S' = s|B' = b]Pr[B' = b] \\
&= Pr[W' = w|B' = b, S' = s]Pr[T' = s - bx'|B' = b]Pr[B' = b] \\
&= Pr[W' = w|B' = b, S' = s]Pr[T' = s - bx']Pr[B' = b].
\end{aligned}$$

Again, the second and the third equalities are conditional probability formulae. The fourth equality is true since $s = t' + bx'$. The last is true because the choice of T' is independent of B'.

Note that:

$$Pr[W = w|B = b, S = s] = \begin{cases} 1 & \text{if } w = h^s y^b \mod n; \\ 0 & \text{otherwise}; \end{cases}$$

$$Pr[W' = w|B' = b, S' = s] = \begin{cases} 1 & \text{if } w = h^s y^b \mod n; \\ 0 & \text{otherwise}; \end{cases}$$

and

$$Pr[T = s - bx] = \begin{cases} 2^{-1.75\ell} & \text{if } 0 < s - bx \le 2^{1.75\ell}; \\ 0 & \text{otherwise;} \end{cases}$$

$$Pr[T' = s - bx'] = \begin{cases} 2^{-1.75\ell} & \text{if } 0 < s - bx' \le 2^{1.75\ell}; \\ 0 & \text{otherwise.} \end{cases}$$

Since B and B' are independently and identically distributed, we have $Pr[B = b] = Pr[B' = b]$. We can now compute $(*)$:

$$\begin{aligned}(*) &= \sum_b \sum_s |Pr[T = s - bx]Pr[B = b] - Pr[T' = s - bx']Pr[B' = b]| \\ &= \sum_b \sum_s Pr[B = b]|Pr[T = s - bx] - Pr[T' = s - bx']|.\end{aligned}$$

Note that $|Pr[T = s - bx] - Pr[T' = s - bx']|$ is non-zero if and only if $s \in [bx, bx'] \cup [2^{1.75\ell} + bx, 2^{1.75\ell} + bx']$. Hence

$$\begin{aligned}(*) &= \sum_b Pr[B = b] \sum_s |Pr[T = s - bx] - Pr[T' = s - bx']| \\ &= \sum_b Pr[B = b] \cdot 2 \cdot b \cdot |x - x'| \cdot 2^{-1.75\ell} \\ &\le \sum_b 2^{-k} \cdot 2 \cdot 2^k \cdot 2^\ell \cdot 2^{-1.75\ell} \\ &\le 2^{k+1+\ell-1.75\ell} \\ &\le 2^{-0.5k+1}.\end{aligned}$$

For the last inequality, we take $\ell \ge 2k$ as defined in Definition 3.1.3. A larger ℓ will make the two distributions more indistinguishable as long as the randomness t used in the signature scheme is chosen from a larger set than the set from which the private key x is chosen. Referring to Definition 3.1.3, we have t chosen from $\mathbb{Z}_{2^{1.75\ell}}$ and x chosen from $\mathbb{Z}_{2^\ell}$, and $\mathbb{Z}_{2^{1.75\ell}}$ is indeed a larger set than $\mathbb{Z}_{2^\ell}$.

To conclude, we have found a required simulation. Applying the forking lemma, i.e., Theorem 3.2.3, we can produce two valid signatures (m, w, b_0, s_0) and (m, w, b_1, s_1) such that $b_0 \ne b_1$ in expected time $T' \le 120686QT/\epsilon$. Note that $s_0 = t + b_0x$ and $s_1 = t + b_1x$. Subtracting s_1 from s_0, and since $b_0 \ne b_1$, we get $x = \frac{s_1 - s_0}{b_1 - b_0}$. Now we can think of the challenger and the attacker as two PPT algorithms and use them as subroutines of an algorithm to solve the discrete logarithm problem in a subgroup of a composite-order group, namely:

1. Input: $n, h, h^{-x} \mod n$.
2. Forge: The attacker forges a signature (m, w, b_0, s_0).
3. Fork: Fork another signature (m, w, b_1, s_1) by the forking lemma. Compute x as described above.
4. Output: x.

Therefore the discrete logarithm in a subgroup of composite-order group can be solved within expected time less than $120686QT/\epsilon$. □

3.3 The Security Parameter

We now briefly discuss how to determine the security parameter k. Suppose that the expected time to solve the composite discrete logarithm problem is $T_{DLP}(k)$, where k is the size of the input of the problem. It is clear that the larger k, the larger T_{DLP}. We want to choose k such that T_{DLP} is larger enough, i.e., larger than $120686QT/\epsilon$, where T could be the length of validity of signatures produced by the Zheng signature scheme, Q is determined depending on different scenarios, and ϵ is the inverse of a chosen polynomial of k.

To rephrase, if there is an attacker against the Zheng signature scheme who can forge a signature within time T with probability ϵ, then according to Theorem 3.2.4, we can produce an algorithm that can solve the composite discrete logarithm problem within time $T' \leq 120686QT/\epsilon$. However, we have chosen k such that the expected time to solve the composite discrete logarithm problem is $T_{DLP} > 120686QT/\epsilon$. Hence, the contradiction shows such an attacker does not exist.

As ϵ is the inverse of a polynomial in k, $120686QT/\epsilon$ is polynomial in k. Note that we have assumed the hardness of the composite discrete logarithm problem. Therefore, there is a value of k such that $T_{DLP} > 120686QT/\epsilon$.

3.4 Summary

In this chapter we provided a formal security proof for the Zheng signature scheme, and showed that the Zheng signature scheme is UF-CMA secure assuming the intractability of the discrete logarithm problem in a composite-order group by applying the forking lemma [30]. The technical difficulty that had to be overcome in this proof was to find a simulation of the signature triple (w, b, s) that has indistinguishable distribution without using the private key. We got around this problem by introducing a fake private key, and argued that the adversary will not notice the difference. Thus, we provided a security proof for a signature scheme that had no previous proof of security.

Chapter 4

Meta-ElGamal Signcryption Schemes

This chapter investigates the meta-ElGamal signature schemes proposed by Horster *et al.* [23]. We give two generic transforms from a meta-ElGamal signature scheme into a signcryption scheme. These constructions generalise the signcryption scheme by Zheng [37] and the signcryption scheme by Gamage *et al.* [18]. Our results show that a large class of signcryption schemes are provably secure. As a by-product, we also show that a large subset of meta-ElGamal signature schemes are secure.

We start with an introduction to the meta-ElGamal signature schemes. We then give two transforms from a meta-ElGamal signature scheme to a signcryption scheme: the Zheng transform and the GLZ transform. We first prove that the signcryption schemes from the Zheng transform have outsider confidentiality and insider unforgeability. We then apply this result to show that the meta-ElGamal signature schemes are UF-CMA secure. Finally we prove that the GLZ transform also provides outsider confidential and insider unforgeable signcryption schemes, where insider unforgeability is proved by using the result that the meta-ElGamal signature schemes are UF-CMA secure.

4.1 Meta-ElGamal Signature Scheme

The ElGamal signature scheme was proposed by ElGamal in 1984 [17]. The security of the scheme relies on the difficulty of computing discrete logarithms over finite fields. However, there was no formal security proof for the ElGamal signature scheme until 2000, when the ElGamal signature scheme with a slight modification was proven formally to be existential unforgeable under an adaptive chosen-message attack in the random oracle model assuming the hardness of the discrete logarithm problem in a prime-order group [30]. The proof is again an application of the forking lemma.

There are many variants of the ElGamal signature scheme, including the Schnorr signature scheme and the Zheng signature scheme. Horster *et al.* [23] integrated these variants and many possible generalisations of the ElGamal signature scheme into a so-called meta-ElGamal signature scheme. As a result, Horster *et al.* obtained more than $13,000$ variants of the ElGamal signature scheme.

We are going to re-define the meta-ElGamal signature scheme below to make the definition of transforms that we propose and the security proofs for the resulting signcryption schemes more compact and concise. Our definition is slightly simpler than the definition in [23], but it covers most of the 13,000 variants identified by Horster *et al.*.

Definition 4.1.1 (Meta-ElGamal Signature Scheme)**.** *A* meta-ElGamal signature scheme *is a signature scheme of a specific form using a group $\mathbb{G}$ generated by an element g of prime order q (where q is a k-bit prime). The key generation algorithm chooses a private key $x \xleftarrow{\$} \mathbb{Z}_q$ and outputs a public key $y \leftarrow g^x$. The scheme is parameterised by three functions $B_1(r, e, s)$, $B_2(r, e, s)$ and $B_3(r, e, s)$, and makes use of a hash function H_S. We will sometimes abbreviate $B_i(r, e, s)$ as B_i for simplicity for $i = 1, 2, 3$. The signature algorithm and verification algorithm work as follows:*

```
Sign(x, m):
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $r \leftarrow \mathcal{E}(w)$
  $e \leftarrow H_S(m, w)$
  Solve $B_1 = xB_2 + tB_3 \mod q$
    for the variable $s$
  $\sigma \leftarrow (w, s)$
  Output $\sigma$
```

```
Verify(y, m, σ):
  Parse $\sigma$ as $(w, s)$
  $r \leftarrow \mathcal{E}(w)$
  $e \leftarrow H_S(m, w)$
  If $g^{B_1} = y^{B_2} w^{B_3}$
    Output $\top$
  Else
    Output $\perp$
```

where $\mathcal{E} : \mathbb{G} \to \mathbb{Z}_q$ *is some encoding function of the group* $\mathbb{G}$ *onto* $\mathbb{Z}_q$.

In our later analysis, we will replace $\mathcal{E}$ with a hash function modeled as a random oracle. Note that r and e are computable from (m, σ). We will concentrate on signature schemes for which

$$B_1, B_2, B_3 \in \{(-1)^{a_0} r^{a_1} e^{a_2} s^{a_3} \; : \; a_0 \in \{0, 1\}, a_1, a_2, a_3 \in \{-1, 0, 1\}\}.$$

Obviously, not all of the possible choices for B_1, B_2, B_3 give rise to secure (or even functionally viable) signature schemes. Most notably, the variable s must exist in at least one clause or the signature scheme cannot be realised. Similarly, the variable e must occur in at least one clause or the signature is trivially insecure.

Both the Zheng [37] and Gamage *et al.* [18] signcryption scheme are essentially based on the *shortened digital signature scheme* (SDSS). The SDSS is a meta-ElGamal scheme defined by $B_1(r, e, s) = es$, $B_2(r, e, s) = -s$ and $B_3(r, e, s) = 1$. Thus, a signature on a message m is a pair (w, s) such that $g^{es} y^s = w$ where $e \leftarrow H_S(m, w)$. We explain how the SDSS is converted into a signcryption scheme in the next section.

4.2 The Two Signcryption Schemes

We first quote the descriptions of the two common signcryption schemes that have inspired the transforms from [15] in Figure 4.1 and Figure 4.2.

```
$Setup(1^k)$
  Pick a random large prime $p$ with
    a $k$-bit prime $q$ dividing $p-1$
  Pick $g \in \mathbb{Z}_p^*$ of order $q$
  Choose a one-time symmetric-key
    encryption scheme
    $SE = (\mathcal{E}_{SE}, \mathcal{D}_{SE})$
    with keyspace $\mathcal{K}$
  Pick two hash functions
    $G : \{0,1\}^* \to \mathcal{K}$
    $H : \{0,1\}^* \to \mathbb{Z}_q$
  $param \leftarrow (p, q, g, SE, G, H)$
  Return $param$
```

```
$KG_S(param)$
  $x_S \xleftarrow{\$} \mathbb{Z}_q$
  $y_S \leftarrow g^{x_S}$
  Return $(x_S, y_S)$
```

```
$KG_R(param)$
  $x_R \xleftarrow{\$} \mathbb{Z}_q$
  $y_R \leftarrow g^{x_R}$
  Return $(x_R, y_R)$
```

```
$SC(param, x_S, y_R, m)$
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $K \leftarrow {y_R}^t$
  $\tau \leftarrow G(K)$
  $e \leftarrow H(m \| y_S \| y_R \| K)$ †
  $c \leftarrow \mathcal{E}_{SE}(\tau, m)$
  $s \leftarrow t/(e + x_S) \mod q$
  $C \leftarrow (c, e, s)$
  Return $C$
```

```
$USC(param, y_S, s_R, C)$
  Parse $C$ as $(c, e, s)$
  $w \leftarrow (y_S g^e)^s$
  $K \leftarrow w^{x_R}$
  $\tau \leftarrow G(K)$
  $m \leftarrow \mathcal{D}_{SE\tau}(c)$
  If $H(m \| y_S \| y_R \| K) = e$
    then return $m$
  Else return $\perp$
```

Figure 4.1: The Zheng signcryption scheme

```
$Setup(1^k)$
  Pick a random large prime $p$ with
    a $k$-bit prime $q$ dividing $p-1$
  Pick $g \in \mathbb{Z}_p^*$ of order $q$
  Choose a one-time symmetric-key
    encryption scheme
    $SE = (\mathcal{E}_{SE}, \mathcal{D}_{SE})$
    with keyspace $\mathcal{K}$
  Pick two hash functions
    $G : \{0,1\}^* \to \mathcal{K}$
    $H : \{0,1\}^* \to \mathbb{Z}_q$
  $param \leftarrow (p, q, g, SE, G, H)$
  Return $param$
```

```
$KG_S(param)$
  $x_S \overset{\$}{\leftarrow} \mathbb{Z}_q$
  $y_S \leftarrow g^{x_S}$
  Return $(x_S, y_S)$
```

```
$KG_R(param)$
  $x_R \overset{\$}{\leftarrow} \mathbb{Z}_q$
  $y_R \leftarrow g^{x_R}$
  Return $(x_R, y_R)$
```

```
$SC(param, x_S, y_R, m)$
  $t \overset{\$}{\leftarrow} \mathbb{Z}_q$
  $K \leftarrow y_R^t$
  $w \leftarrow g^t$
  $\tau \leftarrow G(K)$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow H(c \| y_S \| y_R \| w)$ †
  $s \leftarrow t/(e + x_S) \mod q$
  $C \leftarrow (c, e, s)$
  Return $C$
```

```
$USC(param, y_S, s_R, C)$
  Parse $C$ as $(c, e, s)$
  $w \leftarrow (y_S g^e)^s$
  $K \leftarrow w^{x_R}$
  $\tau \leftarrow G(K)$
  $m \leftarrow \mathcal{D}_{SE_\tau}(c)$
  If $H(c \| y_S \| y_R \| w) = e$
    then return $m$
  Else return $\perp$
```

Figure 4.2: The Gamage *et al.* signcryption scheme

As shown in Figure 4.1 and Figure 4.2, the Gamage *et al.* signcryption scheme is very similar to the Zheng signcryption scheme. The main difference between these two signcryption schemes is highlighted by †. In the Gamage *et al.* signcryption scheme, e is computed as $H(c\|y_S\|y_R\|w)$ instead of $H(m\|y_S\|y_R\|K)$. This change allows public verifiability of the authenticity of the ciphertexts produced by the Gamage *et al.* signcryption scheme, while the ciphertexts produced by the Zheng signcryption scheme can only be verified by the receiver.

4.3 The Properties required on (B_1, B_2, B_3)

As shown in Definition 4.1.1, a meta-ElGamal signature scheme is parameterised by a tuple of functions $(B_1(r,e,s), B_2(r,e,s), B_3(r,e,s))$. For simplicity, we first examine meta-ElGamal signature schemes with $B_3(r,e,s) = 1$. We will discuss the extension to more general signature schemes in Section 4.8. There are four properties that we require on $B_1(r,e,s)$ and $B_2(r,e,s)$ for our transforms to produce secure signcryption schemes:

1. For all $(r,e,x,t) \in \mathbb{Z}_q^4$ there exists a unique solution $s \in \mathbb{Z}_q$ such that $B_1(r,e,s) = xB_2(r,e,s) + t \bmod q$ and that there exists a polynomial-time algorithm to find it. (This allows the signcryption algorithms to find s in polynomial time.)

2. For $(\beta_1, \beta_2, r, e, s) \in \mathbb{Z}_q^5$, fix any three of the five values. Then the other two values are determined by the equations $B_1(r,e,s) = \beta_1$ and $B_2(r,e,s) = \beta_2$, and there exists a polynomial-time algorithm to find them. (This is required for the unsigncryption algorithm and the security proofs.)

3. If at least one of r,e,s is uniformly random over $\mathbb{Z}_q$ and the others are fixed values, then $B_1(r,e,s)$ and $B_2(r,e,s)$ have a joint distribution that

is computationally indistinguishable from a uniform distribution over $\mathbb{Z}_q \times \mathbb{Z}_q$. (To simplify the security proofs a little bit, we will assume the distribution is exactly uniform.)

4. For $(r, r', e, e', s, s') \in \mathbb{Z}_q^6$, if $B_1(r, e, s) = B_1(r', e', s')$ and $B_2(r, e, s) = B_2(r', e', s')$ then $e = e'$.

The third and the fourth properties are required for the security proofs of signcryption schemes obtained from the transforms.

4.4 The Two Transforms

Our two transforms which convert a meta-ElGamal signature scheme into a meta-ElGamal signcryption scheme are based on the Zheng signcryption scheme [37] and the Gamage *et al.* signcryption scheme [18] respectively. The construction essentially re-uses the random element $w = g^t$ computed as part of the meta-ElGamal signature scheme as the ephemeral public key in a Diffie-Hellman key exchange [16]. It can be viewed as the combination of a meta-ElGamal signature with a symmetric encryption scheme [1].

For both transforms, the *Setup* algorithm generates a group $\mathbb{G}$ with a generator g of prime order q (where q is a k-bit prime). The sender key generation algorithm KG_S chooses a private key $x_S \xleftarrow{\$} \mathbb{Z}_q$ and computes a public key $y_S \leftarrow g^{x_S}$. The receiver key generation algorithm KG_R also chooses a private key $x_R \xleftarrow{\$} \mathbb{Z}_q$ and computes a public key $y_R \leftarrow g^{x_R}$. The sender key generation process can be thought of as the key generation algorithm for the meta-ElGamal signature scheme and the receiver key generation process can be thought of as the key generation algorithm for the encryption scheme; hence, although the processes are functionally identical, they derive from different conceptual roots.

The signcryption/unsigncryption algorithms for the Zheng transform and the GLZ transform are given in Figure 4.3 and Figure 4.4. The schemes make

```
$SC(x_S, y_R, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $K \leftarrow y_R^t$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow H_1(m, y_S, y_R, K)$
  $r \leftarrow H_2(w)$
  Solve $B_1(r, e, s) = x_S B_2(r, e, s) + t$
    for the variable $s$
  $C \leftarrow (c, B_1(r, e, s), B_2(r, e, s))$
  Output $C$
```

```
$USC(y_S, x_R, C)$
  Parse $C$ as $(c, \beta_1, \beta_2)$
  $w \leftarrow g^{\beta_1} y_S^{-\beta_2}$
  $r \leftarrow H_2(w)$
  Compute $(e, s)$ from $(\beta_1, \beta_2, r)$ using
    $B_1(r, e, s) = \beta_1$ and $B_2(r, e, s) = \beta_2$
  $K \leftarrow w^{x_R}$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $m \leftarrow \mathcal{D}_{SE\tau}(c)$
  If $H_1(m, y_S, y_R, K) \neq e$
    output $\perp$
  Else output $m$
```

Figure 4.3: The Zheng transform of a meta-ElGamal signature scheme into a meta-ElGamal signcryption scheme

use of a symmetric encryption scheme $(\mathcal{E}_{SE}, \mathcal{D}_{SE})$ with keyspace $\{0,1\}^\ell$ and a set of hash functions:

$$H_1 : \{0,1\}^* \times \mathbb{G}^3 \to \mathbb{Z}_q \qquad H_2 : \mathbb{G} \to \mathbb{Z}_q \qquad H_3 : \mathbb{G}^3 \to \{0,1\}^\ell .$$

Recall that the SDSS has $B_1(r, e, s) = es$ and $B_2(r, e, s) = -s$. If we apply the Zheng transform to the SDSS, we get a signcryption scheme which outputs ciphertexts of the form $(c, et/(e + x_S), -t/(e + x_S))$. This is essentially equivalent to the provably-secure version of Zheng's signcryption scheme [4]. Similarly, if we apply the GLZ transform to the SDSS, we get an equivalent version of the Gamage *et al.* signcryption scheme [18].

4.5 Notation for the Security Proofs

The security proofs require a complex interaction of the random oracle simulations. We keep track of these queries and responses using a series of lists L^{H_1}, L^{H_2} and L^{H_3}. We count each query that an adversary $\mathcal{A}$ makes to any oracle and index the lists accordingly; so, for example, if the i-th oracle query made by $\mathcal{A}$ is to the H_1 oracle, then an entry is added to L^{H_1} indexed by i.

```
$SC(x_S, y_R, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $K \leftarrow y_R^t$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow H_1(c, y_S, y_R, w)$
  $r \leftarrow H_2(w)$
  Solve $B_1(r,e,s) = x_S B_2(r,e,s) + t$
    for the variable $s$
  $C \leftarrow (c, B_1(r,e,s), B_2(r,e,s))$
  Output $C$
```

```
$USC(y_S, x_R, C)$
  Parse $C$ as $(c, \beta_1, \beta_2)$
  $w \leftarrow g^{\beta_1} y_S^{-\beta_2}$
  $r \leftarrow H_2(w)$
  Compute $(e, s)$ from $(\beta_1, \beta_2, r)$ using
    $B_1(r,e,s) = \beta_1$ and $B_2(r,e,s) = \beta_2$
  If $H_1(c, y_S, y_R, w) \neq e$ then output $\perp$
  $K \leftarrow w^{x_R}$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $m \leftarrow \mathcal{D}_{SE\tau}(c)$
  Output $m$
```

Figure 4.4: The GLZ transform of a meta-ElGamal signature scheme into a meta-ElGamal signcryption scheme

This does not imply that there exists an $(i-1)$-th entry on the list L^{H_1} as the $(i-1)$-th query may not have been to the oracle H_1. Signcryption and unsigncryption oracle queries may update multiple lists simultaneously. We can therefore associate an index set I^{H_1} with the list L^{H_1} which contains the set of indices for which there exists an entry on L^{H_1} (and similarly for the other oracles).

We will describe the contents of each list when they are formally introduced (in the security proof). However, it is useful to introduce some extra notation at this point. If X is a set then we define $\overline{X} = X \cup \{\star\}$. An asterisk $\star$ will denote an entry in a list which could not be computed by the simulation. We let $\cdot$ denote a symbol which "matches" any entry. Thus, $(x, y, \cdot) = (u, v, w)$ if and only if $x = u$ and $y = v$. This allows us to make statements such as "there exists $(x, y, \cdot) = (x_i, y_i, z_i) \in L^X$ for some $i \in I^X$", which means that there exists an index i (corresponding to the i-th oracle query) such that $x = x_i$ and $y = y_i$. Our simulations will be such that there exists at most one entry $i \in I^X$ such that $(x, y, \cdot) = (x_i, y_i, z_i) \in L^X$.

4.6 Confidentiality of the Zheng Transform

In this section, we will prove the confidentiality of the Zheng transform.

The confidentiality of our schemes obtained from the Zheng transform relies on the IND-OT security of the symmetric encryption scheme (Definition 2.3.2) and the gap Diffie-Hellman problem in the group $\mathbb{G}$ (Definition 2.2.8).

Theorem 4.6.1. *Let SC be a signcryption scheme obtained from the Zheng transform. Given a PPT adversary $\mathcal{A}$ against the IND-CCA2 property of SC, there exist a PPT adversary $\mathcal{B}$ against the GDH problem and a PPT adversary $\mathcal{B}'$ against the IND-OT property of the symmetric encryption scheme such that*

$$Adv_{\mathcal{A}}^{\mathtt{out\text{-}IND}}(k) \leq 2q_S\frac{q_1+q_2+2q_S+2q_U}{q} + 2\frac{q_S+q_U}{q} + 2Adv_{\mathcal{B}}^{\mathtt{GDH}}(k) + Adv_{\mathcal{B}'}^{\mathtt{PA}}(k)\,,$$

where q_i is the maximum number of queries made by $\mathcal{A}$ to the H_i oracle, q_S is the maximum number of queries made to the signcryption oracle, and q_U is the maximum number of queries made to the unsigncryption oracle.

Proof: Let $\mathcal{A}$ be a PPT adversary against the IND-CCA2 property of the signcryption scheme. We will prove the theorem via a sequence of games [8, 34]. Each game will be parameterised by a bit b and we define W_i^b to be the event that $\mathcal{A}$ outputs 1 in the version of game G_i which uses bit b.

Game G_1:

G_1 is the game $\text{Expt}_{\mathcal{A}}^{\mathtt{out\text{-}IND\text{-}b}}(k)$ defined in Definition 2.3.8. Hence:

$$Adv_{\mathcal{A}}^{\mathtt{out\text{-}IND}}(k) = |\Pr[W_1^1] - \Pr[W_1^0]|\,.$$

Game G_2:

G_2 simulates the random oracles, the signcryption oracle and the unsigncryption oracle internally using a series of lists. The lists L^{H_1}, L^{H_2}, L^{H_3} contain elements of the form $(m, y_S, y_R, w, K, e) \in \{0,1\}^* \times \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \mathbb{Z}_q$, $(w, r) \in \mathbb{G} \times \mathbb{Z}_q$, $(y_S, y_R, w, K, \tau) \in \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \{0,1\}^\ell$ corresponding to the input/output of the hash functions. Recall that $\mathcal{O}$ is a DDH oracle which determines whether (a, b, c) satisfies $\log_g b = \log_a c$.

We will change the way in which the oracles work, but first we need to consider how to "address" elements of the lists L^{H_1} and L^{H_3}. As an example, we consider the list L^{H_1}. Elements of this list will always have the form $(m, y, y', \star, K, e)$ or $(m, y, y', w, \star, e)$. A query $H_1(m, y, y', K)$ should receive the answer e if either $(m, y, y', \star, K, e) \in L^{H_1}$ or if $(m, y, y', w, \star, e) \in L^{H_1}$ where $w = g^t$ and $K = y'^t$ for some t. We define the *Def*/*Def*$'$ functions in Figure 4.5 to easily determine whether an appropriate entry exists in the list.

This allows us to re-define the hash/signcryption/unsigncryption oracles as in Figure 4.6.

We assume that the challenge ciphertext is computed as $\mathcal{O}_S(y_R^*, m_b)$. Since the hash functions are modelled as random oracles, and the lists ensure the consistency among the hash queries, we have that G_2 is equivalent to G_1. Hence,

$$\Pr[W_2^b] = \Pr[W_1^b] \quad \text{where } b \in \{0, 1\} \ .$$

Game G_3:

Note that in G_2, the receiver's private key x_R^* is not used at all. G_3 changes the action of the signcryption oracle so that it computes signcryption ciphertexts without using the sender's private key x_S^*. This is done by changing the signcryption oracle $\mathcal{O}_S$ to act as in Figure 4.7. The challenge ciphertext is then computed as $\mathcal{O}_S(y_R^*, m_b)$.

Since $r, e, s \xleftarrow{\$} \mathbb{Z}_q$, we have that $B_1(r, e, s)$ and $B_2(r, e, s)$ have a uniform joint distribution over $\mathbb{Z}_q \times \mathbb{Z}_q$ according to the third property listed in Section 4.3. As w is computed as $g^{B_1} y_S^{*B_2}$, w is therefore uniformly distributed over $\mathbb{G}$. Thus, as long as $\perp_1$ and $\perp_2$ do not occur, we have that G_2 and G_3 are equivalent. Note that the size of L^{H_i} is bounded by $q_i + q_S + q_U$ for $i \in \{1, 2\}$. Since w is distributed at random over $\mathbb{G}$, we have that the probability $\perp_1$ occurs in any execution of $\mathcal{O}_S$ is bounded by $(q_1 + q_S + q_U)/q$ and

```
$Def_1(m, y, y', K)$:
  If $(m, y, y', \cdot, K, \cdot) = (m_i, y_i, y'_i, w_i, K_i, e_i)$ for $i \in I^{H_1}$
    $e \leftarrow e_i$
  Else
    if $(m, y, y', \cdot, \star, \cdot) = (m_i, y_i, y'_i, w_i, K_i, e_i)$ and $\mathcal{O}(w_i, y', K) = 1$ for $i \in I^{H_1}$
      $e \leftarrow e_i$
    Else
      $e \leftarrow \perp$
  Return $e$

$Def'_1(m, y, y', w)$:
  If $(m, y, y', w, \star, \cdot) = (m_i, y_i, y'_i, w_i, K_i, e_i)$ for $i \in I^{H_1}$
    $e \leftarrow e_i$
  Else
    if $(m, y, y', \star, \cdot, \cdot) = (m_i, y_i, y'_i, w_i, K_i, e_i)$ and $\mathcal{O}(w, y', K_i) = 1$ for $i \in I^{H_1}$
      $e \leftarrow e_i$
    Else
      $e \leftarrow \perp$
  Return $e$

$Def_2(w)$:
  If $(w, \cdot) = (w_i, r_i)$ for some $i \in I^{H_2}$
    $r \leftarrow r_i$
  Else
    $r \leftarrow \perp$
  Return $r$

$Def_3(y, y', K)$:
  If $(y, y', \cdot, K, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else if $(y, y', \cdot, \star, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ and $\mathcal{O}(y', w_i, K) = 1$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else
    $\tau \leftarrow \perp$
  Return $\tau$

$Def'_3(y, y', w)$:
  If $(y, y', w, \star, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else
    if $(y, y', \star, \cdot, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ and $\mathcal{O}(y', w, K_i) = 1$ for $i \in I^{H_3}$
      $\tau \leftarrow \tau_i$
    Else
      $\tau \leftarrow \perp$
  Return $\tau$
```

Figure 4.5: Functions which determine membership of the lists L^{H_1}, L^{H_2} and L^{H_3} from partial information

```
$H_1(m, y, y', K)$:
  $e \leftarrow Def_1(m, y, y', K)$
  If $e = \perp$
    $e \xleftarrow{\$} \mathbb{Z}_q$
    Add $(m, y, y', \star, K, e)$
      to $L^{H_1}$
  Return $e$
```

```
$H_2(w)$:
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$
      to $L^{H_2}$
  Return $r$
```

```
$H_3(y, y', K)$:
  $\tau \leftarrow Def_3(y, y', K)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y, y', \star, K, \tau)$
      to $L^{H_3}$
  Return $\tau$
```

```
$\mathcal{O}_S(y_R, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $\tau \leftarrow Def'_3(y_S^*, y_R, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S^*, y_R, w, \star, \tau)$ to $L^{H_3}$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow Def'_1(m, y_S^*, y_R, w)$
  If $e = \perp$
    $e \xleftarrow{\$} \mathbb{Z}_q$
    Add $(m, y_S^*, y_R, w, \star, e)$ to $L^{H_1}$
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$ to $L^{H_2}$
  Solve $B_1(r, e, s) = x_S^* B_2(r, e, s) + t$
    for the variable $s$
  $C \leftarrow (c, B_1(r, e, s), B_2(r, e, s))$
  Return $C$
```

```
$\mathcal{O}_U(y_S, C)$:
  Parse $C$ as $(c, B_1, B_2)$
  $w \leftarrow g^{B_1} y_S^{B_2}$
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$ to $L^{H_2}$
  Compute $(e, s)$ from $(B_1, B_2, r)$
  $\tau \leftarrow Def'_3(y_S, y_R^*, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S, y_R^*, w, \star, \tau)$ to $L^{H_3}$
  $m \leftarrow \mathcal{D}_{SE\tau}(c)$
  $e' \leftarrow Def'_1(m, y_S, y_R^*, w)$
  If $e' = \perp$
    $e' \xleftarrow{\$} \mathbb{Z}_q$
    Add $(m, y_S, y_R^*, w, \star, e')$ to $L^{H_1}$
  If $e \neq e'$ then return $\perp$
  Else return $m$
```

Figure 4.6: The hash/signcryption/unsigncryption oracles in Game G_2

```
$\mathcal{O}_S(y_R, m)$:
  $r, e, s \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^{B_1(r,e,s)} y_S^{*B_2(r,e,s)}$
  $\tau \leftarrow Def'_3(y_S^*, y_R, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S^*, y_R, w, \star, \tau)$ to $L^{H_3}$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e' \leftarrow Def'_1(m, y_S^*, y_R, w)$
  If $e' \neq \perp$
    Output $\perp$ and halt all operations                $\triangleright$We call this event $\perp_1$
  Add $(m, y_S^*, y_R, w, \star, e)$ to $L^{H_1}$
  $r' \leftarrow Def_2(w)$
  If $r' \neq \perp$
    Output $\perp$ and halt all operations                $\triangleright$We call this event $\perp_2$
  Add $(w, r)$ to $L^{H_2}$
  $C \leftarrow (c, B_1(r,e,s), B_2(r,e,s))$
  Return $C$
```

Figure 4.7: The signcryption oracle in Game G_3

there are q_S executions. Similarly, the probability that $\perp_2$ occurs is bounded by $(q_2 + q_S + q_U)/q$ in any execution of $\mathcal{O}_S$. Thus,

$$|\Pr[W_3^b] - \Pr[W_2^b]| \leq q_S \cdot \left(\frac{q_1 + q_2 + 2q_S + 2q_U}{q} \right) .$$

Game G_4:

G_4 is identical to G_3 except for two tiny changes which ensure that the hash oracles only evaluate $H_1(m_b, y_S^*, y_R^*, K^*)$ or $H_3(y_S^*, y_R^*, K^*)$ during the computation of the challenge ciphertext. Formally, we can break this down into three cases:

1. The adversary could make a direct hash oracle query $H_3(y_S^*, y_R^*, K^*)$ or $H_1(m, y_S^*, y_R^*, K^*)$ for any m. This allows us to recover K^*. We define this to be event E_1. As we shall see, this leads to an algorithm which solves the GDH problem. Note that this event can be detected in polynomial-time using the DDH oracle.

2. The signcryption oracle $\mathcal{O}_S$ could attempt to make such a query; however, in that case, the value w computed by the signcryption oracle must be equal to w^*. Since both of these values have uniform distribution, the probability that this occurs is bounded by q_S/q.

3. The unsigncryption oracle $\mathcal{O}_U$ could attempt to make such a query; however, in that case, the adversary must have submitted a query (c, B_1, B_2) such that $g^{B_1} y_S^{*B_2} = g^{B_1^*} y_S^{*B_2^*}$. This is split into two cases:

   - If $(B_1, B_2) = (B_1^*, B_2^*)$ then we must have $c \neq c^*$. We must have that $\tau = \tau^*$ and since the symmetric encryption scheme is one-to-one we must have that $m \neq m_b$. We must also have $e = e^*$. Therefore, we have that $H_1(m, y_S^*, y_R^*, K^*) = e^*$ with probability $1/q$, as $\mathcal{A}$ cannot have made any direct $H_1(m, y_S^*, y_R^*, K^*)$ queries. We therefore alter the unsigncryption oracle so that it outputs $\perp$

```
$\mathcal{B}^{\mathcal{O}}(g, X, Y)$:
  $w^* \leftarrow X$
  $y_R^* \leftarrow Y$
  $r^*, e^*, s^* \xleftarrow{\$} \mathbb{Z}_q$
  Add $(w^*, r^*)$ to $L^{H_2}$
  $\tau^* \leftarrow \{0,1\}^\ell$
  $y_S^* \leftarrow (g^{B_1(r^*,e^*,s^*)} w^{*-1})^{1/B_2(r^*,e^*,s^*)}$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1(y_S^*, y_R^*)$
  $c^* \leftarrow \mathcal{E}_{SE_{\tau^*}}(m_b)$
  $C^* \leftarrow (c^*, B_1^*(r^*, e^*, s^*), B_2^*(r^*, e^*, s^*))$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  If $E_1$ or $E_2$ occurred then output $K^*$
  Else output $\perp$
```

Figure 4.8: The algorithm for solving the GDH problem

if queried on (c, B_1^*, B_2^*). The probability that this is incorrect is bounded by q_U/q.

- If $(B_1, B_2) \neq (B_1^*, B_2^*)$ then we must have $B_1^* \neq B_1$ and $B_2^* \neq B_2$. So we may recover t_S^* as $(B_1^* - B_1)/(B_2 - B_2^*)$ and recover K^* as ${y_R^*}^{t^*}$. We define this to be event E_2. As we shall see, this leads to an algorithm which solves the GDH problem. Note that this event can be detected in polynomial time by computing w.

We define an adversary $\mathcal{B}$ which solves the GDH problem if E_1 or E_2 occurs in Figure 4.8.

If $\mathcal{A}$ makes any oracle queries, then they are answered using the simulations of H_1, H_2, H_3, $\mathcal{O}_S$ and $\mathcal{O}_U$ (with the exception that $\mathcal{O}_U$ returns $\perp$ on queries of the form (c, B_1^*, B_2^*)). Thus,

$$|\Pr[W_4^b] - \Pr[W_3^b]| \leq \frac{q_S + q_U}{q} + Adv_{\mathcal{B}}^{\texttt{GDH}}(k)\,.$$

So, in G_4, the only time that $\mathcal{A}$ receives any data that depends on τ^* is when it receives c^* as part of the challenge ciphertext. Thus, we can give

```
$\mathcal{B}'_1(1^k)$:                                    $\mathcal{B}'_2(c^*)$:
  $x^*_S, x^*_R \xleftarrow{\$} \mathbb{Z}_q$                         $C^* \leftarrow (c^*, B^*_1, B^*_2)$
  $y^*_S \leftarrow g^{x^*_S}$; $y^*_R \leftarrow g^{x^*_R}$                  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $r^*, e^*, s^* \xleftarrow{\$} \mathbb{Z}_q$                         $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  $B^*_i \leftarrow B_i(r^*, e^*, s^*)$ for $i = 1, 2$          If $\mathcal{A}$ made an "illegal" query
  $w^* \leftarrow g^{B^*_1} y^{*B^*_2}$                             output 0
  Add $(w^*, r^*)$ to $L^{H_2}$                       Else output $b'$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1(y^*_S, y^*_R)$
  Output $(m_0, m_1)$
```

Figure 4.9: The adversary $\mathcal{B}'$ against the IND-OT security of the symmetric encryption scheme

an adversary $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ against the IND-OT security of the symmetric encryption scheme, by implicitly setting $H_3(y^*_R, y^*_R, K^*)$ to be equal to the hidden symmetric key used in the IND-OT game. The adversary $\mathcal{B}'$ runs as shown in Figure 4.9 (we assume that all required variables are passed between $\mathcal{B}'_1$ and $\mathcal{B}'_2$ as part of the state variable).

$\mathcal{A}$'s oracle queries are answered using the simulators H_1, H_2, H_3, $\mathcal{O}_S$ and $\mathcal{O}_U$. An "illegal" oracle query is either a hash query $H_1(m, y^*_S, y^*_R, K^*)$ or $H_3(y^*_S, y^*_R, K^*)$ where $K^* = w^{*x^*_S}$, a signcryption query with $w = w^*$ or an unsigncryption oracle query with $w = w^*$ and $(B_1, B_2) \neq (B^*_1, B^*_2)$. An examination shows that

$$|\Pr[W^1_4] - \Pr[W^0_4]| = Adv^{\texttt{PA}}_{\mathcal{B}'}(k)\,.$$

Thus, we can conclude

$$\begin{aligned} Adv^{\texttt{out-IND}}_{\mathcal{A}}(k) &= |\Pr[W^1_1] - \Pr[W^0_1]| \\ &\leq 2q_S \frac{q_1 + q_2 + 2q_S + 2q_U}{q} + 2\frac{q_S + q_U}{q} + 2Adv^{\texttt{GDH}}_{\mathcal{B}}(k) + Adv^{\texttt{PA}}_{\mathcal{B}'}(k)\,. \end{aligned}$$

□

4.7 Unforgeability of the Zheng Tranform

In this section, we will prove the unforgeability of the Zheng transform. The security proof uses the general forking lemma [6] and has a listing system that is the same as before. The unforgeability of the scheme relies on the gap discrete logarithm problem in the group $\mathbb{G}$ (Definition 2.2.7).

Theorem 4.7.1. *Let SC be a signcryption scheme obtained from the Zheng transform. If there exists a PPT adversary $\mathcal{A}$ against the UF-CMA property of SC, then there exists a PPT adversary $\mathcal{B}$ against the GDL problem such that*

$$Adv_{\mathcal{A}}^{\textit{in-UF}}(k) \leq \sqrt{(q_1+1)Adv_{\mathcal{B}}^{\textit{GDL}}(k) + \frac{(q_1+1)^2}{4q^2}} + \frac{q_1 + 2q_S(q_1+q_2+2q_S+2)+3}{2q}$$

where q_i is the maximum number of queries made by $\mathcal{A}$ to the H_i oracle for $i \in \{1,2\}$ and q_S is the maximum number of queries made to the signcryption oracle.

We prove this theorem using two lemmas. As an intermediate step, we define a mathematical problem that is similar to the GDL problem in Definition 2.2.7.

Definition 4.7.2 (GDL′). *Given $k \in \mathbb{N}$, generate a group $\mathbb{G}$ of prime order q with some generator g. Pick $a \xleftarrow{\$} \mathbb{Z}_q$. The input of the problem is (g, q, g^a). We define an oracle $\mathcal{R}$ as follows: for $i = 1, \ldots, q_R$, on input $(y_i, K_i) \in \mathbb{G}^2$, return $e_i \xleftarrow{\$} \mathbb{Z}_q$. The problem is to find a tuple (w, e_{i^*}, s) for some positive integer $i^* \leq q_R$ such that*

$$\log_g w = \log_{y_{i^*}} K_{i^*} \qquad \textit{and} \qquad K_{i^*} = {y_{i^*}}^{B_1(r,e_{i^*},s) - aB_2(r,e_{i^*},s)},$$

where B_1, B_2 were defined in the signcryption scheme, and $r \leftarrow H_2(w)$.

The first lemma is stated as follows:

Lemma 4.7.3. *If there exists a PPT adversary $\mathcal{A}$ against the UF-CMA property of the signcryption scheme, then there exists a PPT adversary $\mathcal{B}'$ against the GDL$'$ problem such that*

$$Adv_{\mathcal{A}}^{\texttt{in-UF}}(k) \leq Adv_{\mathcal{B}'}^{\texttt{GDL}'}(k) + \frac{q_S(q_1 + q_2 + 2q_S + 2) + 1}{q} .$$

Proof: Let $\mathcal{A}$ be a PPT adversary against the UF-CMA property of the signcryption scheme. We will prove the theorem as before via a sequence of games [8, 34]. At the end of each game, $\mathcal{A}$ will output a tuple consisting of (pk_R^*, sk_R^*, C^*). On input the tuple, a checking algorithm will output 0 or 1 based on the three conditions listed in $\textsc{Expt}_{\mathcal{A}}^{\texttt{in-UF}}(k)$ in Definition 2.3.11. We define W_i to be the event that the checking algorithm outputs 1 in Game i.

Game G_1:

G_1 is the game $\textsc{Expt}_{\mathcal{A}}^{\texttt{in-UF}}(k)$. Hence:

$$\Pr[W_1] = Adv_{\mathcal{A}}^{\texttt{in-UF}}(k) .$$

Game G_2:

G_2 simulates the hash oracles using a series of lists. We will assume that H_1 has access to the oracle $\mathcal{R}$ defined in the GDL$'$ problem. We introduce a new list L^J that contains elements of the form $(y_R, K, e, j) \in \mathbb{G}^2 \times \mathbb{Z}_q \times \mathbb{Z}$ defined by the query/responses of the $\mathcal{R}$ oracle. The value j is assumed to be initially set to 0. We adapt the notation introduced in the last proof as in Figure 4.10.

This game is equivalent to G_1 and so we have:

$$Pr[W_2] = Pr[W_1] .$$

Game G_3:

G_3 simulates the signcryption oracle so that the queries can be answered without the private keys. This is achieved with the help of the DDH oracle $\mathcal{O}$. The simulation is shown in Figure 4.11.

```
$H_1^{\mathcal{R}}(m, y, y', K)$:
  $e \leftarrow \mathit{Def}_1(m, y, y', K)$
  If $e = \perp$
    $j \leftarrow j + 1$
    Query $e \xleftarrow{\$} \mathcal{R}(y_R, K)$
    Add $(y_R, K, e, j)$ to $L^J$
    Add $(m, y, y', \star, K, e)$
      to $L^{H_1}$
  Return $e$
```

```
$H_2(w)$:
  $r \leftarrow \mathit{Def}_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$
      to $L^{H_2}$
  Return $r$
```

```
$H_3(y, y', K)$:
  $\tau \leftarrow \mathit{Def}_3(y, y', K)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^{\ell}$
    Add $(y, y', \star, K, \tau)$
      to $L^{H_3}$
  Return $\tau$
```

Figure 4.10: The hash oracles in Game G_2

```
$\mathcal{O}_S(y_R, m)$:
  $r, e, s \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^{B_1(r,e,s)} {y^*}_S^{-B_2(r,e,s)}$
  $\tau \leftarrow \mathit{Def}'_3(y^*_S, y_R, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^{\ell}$
    Add $(y^*_S, y_R, w, \star, \tau)$ to $L^{H_3}$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e' \leftarrow \mathit{Def}'_1(m, y^*_S, y_R, w)$
  If $e' \neq \perp$
    Output $\perp$ and halt all operations                $\triangleright$We call this event $\perp_1$
  Add $(m, y^*_S, y_R, w, \star, e)$ to $L^{H_1}$
  $r' \leftarrow \mathit{Def}_2(w)$
  If $r' \neq \perp$
    Output $\perp$ and halt all operations                $\triangleright$We call this event $\perp_2$
  Add $(w, r)$ to $L^{H_2}$
  $C \leftarrow (c, B_1(r, e, s), B_2(r, e, s))$
  Return $C$
```

Figure 4.11: The signcryption oracle $\mathcal{O}_S$ in Game G_3

Since $r, e, s \xleftarrow{\$} \mathbb{Z}_q$, we have that w is uniformly distributed over $\mathbb{G}$ (by definition of $B_1(r, e, s)$ and $B_2(r, e, s)$). Thus, as long as $\perp_1$ and $\perp_2$ do not occur, we have that G_2 and G_3 are equivalent. Note that the size of L^{H_i} is bounded by $q_i + q_S + 1$ for $i \in \{1, 2\}$ (the "+1" comes from the checking algorithm). Since w is distributed at random over $\mathbb{G}$, we have that the probability $\perp_1$ occurs in any execution of $\mathcal{O}_S$ is bounded by $(q_1 + q_S + 1)/q$. Similarly, the probability that $\perp_2$ occurs is bounded by $(q_2 + q_S + 1)/q$. Thus,

$$|\Pr[W_3] - \Pr[W_2]| \leq q_S \cdot \left(\frac{q_1 + q_2 + 2q_S + 2}{q} \right) .$$

On the other hand, let us analyze the event W_3 by considering the following two cases:

1. H_1 has been queried on (m^*, y_S^*, y_R^*, K^*). We will call this event *Bad*.

2. H_1 has not been queried on (m^*, y_S^*, y_R^*, K^*). (If C^* is a valid forgery then $H_1(m^*, y_S^*, y_R^*, K^*)$ cannot have been defined by the signcryption oracle.) Then e' is independently uniformly generated from $\mathbb{Z}_q$. So in this case,
$$Pr[W_3 \,|\, \neg Bad] \leq \frac{1}{q}.$$

Hence, we have
$$Pr[W_3] \leq \frac{1}{q} + Pr[Bad].$$

Lastly, we show that $\Pr[Bad]$ can be upper-bounded by $Adv_{\mathcal{B}'}^{\mathtt{GDL}'}(k)$. The adversary $\mathcal{B}'$ takes (g, q, X) as input and runs as follows:

```
$\mathcal{B}'^{\mathcal{O},\mathcal{R}}(g, q, X)$:
  $y_S^* \leftarrow X$
  $(y_R^*, x_R^*, C^*) \leftarrow \mathcal{A}(y_S^*)$
  Parse $C^*$ as $(c^*, {\beta_1}^*, {\beta_2}^*)$
  $w^* \leftarrow g^{{\beta_1}^*} {y_S^*}^{-{\beta_2}^*}$
  $r^* \leftarrow H_2(w^*)$
  Compute $(e^*, s^*)$ from $({\beta_1}^*, {\beta_2}^*, r^*)$
  $K^* \leftarrow w^{* x_R^*}$
  $\tau^* \leftarrow H_3(y_S^*, y_R^*, K^*)$
  $m^* \leftarrow \mathcal{D}_{SE_{\tau^*}}(c^*)$
  $e' \leftarrow H_1(m^*, y_S^*, y_R^*, K^*)$
  If $e' \neq e^*$
    Output $\perp$
  If $(y_{R^*}, K^*, e^*, \cdot) = (y_j, K_j, e_j, j) \in L^J$ for some $1 \le j \le q_1 + 1$
    Output $(w^*, e^*, s^*, j)$
  Else output $\perp$
```

The H_1, H_2, H_3 and $\mathcal{O}_S$ oracles are simulated as in Game G_3.

If $\mathcal{A}$ outputs a successful forgery, then $H_1(m^*, y_S^*, y_R^*, K^*)$ must be defined by a direct H_1-oracle query. Thus there will exist an entry $(y_R^*, K^*, e^*, j) \in L^J$ for some $1 \le j \le q_1 + 1$. Thus, (w^*, e^*, s^*, j) is a valid solution to the GDL$'$ problem and $\Pr[Bad] \le Adv_{\mathcal{B}'}^{\texttt{GDL}'}(k)$. In conclusion,

$$Adv_{\mathcal{A}}^{\texttt{in-UF}}(k) \le Adv_{\mathcal{B}'}^{\texttt{GDL}'}(k) + \frac{q_S(q_1 + q_2 + 2q_S + 2) + 1}{q}.$$

□

As we have proved that the unforgeability of our signcryption scheme relies on the intractability of the GDL$'$ problem, the next task is to link the intractability of the GDL$'$ problem to the intractability of the GDL problem. We will use the general forking lemma from Bellare and Neven [6] to show the link and we state it as follows:

Lemma 4.7.4 (General Forking Lemma [6]). *Fix an integer $q_R \ge 1$ and a set Z of size $q \ge 2$. Let A be a randomized algorithm that on input $(PP, e_1, \ldots, e_{q_R})$ returns a pair, the first element of which is an integer in the range $0, \ldots, q_R$ and the second element of which we refer to as a side output. Let IG be a randomized algorithm that we call the input generator. The* *accepting probability of A, denoted acc, is defined as the probability that $J \geq 1$ in the experiment*

$$\begin{aligned} &PP \xleftarrow{\$} IG \\ &e_1, \ldots, e_{q_R} \xleftarrow{\$} Z \\ &(J, \sigma) \xleftarrow{\$} A(PP, e_1, \ldots, e_{q_R}) \end{aligned}$$

The forking algorithm F_A associated to A is the randomized algorithm that takes input PP and proceeds as follows:

```
$F_A(PP)$:
  Pick coins $\rho$ for $A$ at random
  $e_1, \ldots, e_{q_R} \xleftarrow{\$} Z$
  $(I, \sigma) \leftarrow A(PP, e_1, \ldots, e_{q_R}; \rho)$
  If $I = 0$
    return $(0, \star, \star)$
  $e'_I, \ldots, e'_{q_R} \xleftarrow{\$} Z$
  $(I', \sigma') \leftarrow A(PP, e_1, \ldots, e_{I-1}, e'_I, \ldots, e'_{q_R}; \rho)$
  If $I = I'$ and $e_I \neq e'_I$
    return $(1, \sigma, \sigma')$
  Else return $(0, \star, \star)$
```

Let

$$frk = Pr[b = 1 : PP \xleftarrow{\$} IG;\ (b, \sigma, \sigma') \xleftarrow{\$} F_A(PP)]$$

Then

$$frk \geq acc \cdot \left(\frac{acc}{q_R} - \frac{1}{q} \right).$$

We are now ready to state our second lemma as follows:

Lemma 4.7.5. *If there exists a PPT adversary $\mathcal{B}'$ against the GDL$'$ problem, then there exists a PPT adversary $\mathcal{B}$ against the GDL problem such that*

$$Adv_{\mathcal{B}}^{GDL}(k) \geq Adv_{\mathcal{B}'}^{GDL'}(k)\big(\frac{Adv_{\mathcal{B}'}^{GDL'}(k)}{q_{\mathcal{R}}} - \frac{1}{q}\big) \qquad where \qquad q_{\mathcal{R}} = q_1 + 1\,.$$

Proof: We are going to use the general forking lemma in this proof. One can think of PP as public information and $e_1, \ldots, e_{q_R}$ as replies to queries to a random oracle.

To apply the lemma, we will describe the algorithm A (which would be essentially the same as $\mathcal{B}'$), and then construct the forking algorithm F_A (which will be used by $\mathcal{B}$). Let $PP \leftarrow (g, q, X)$ be the GDL problem instance and $e_i \xleftarrow{\$} \mathbb{Z}_q$ for $i = 1, \ldots, q_R$.

```
$A^{\mathcal{O}}(g, q, X, e_1, \ldots, e_{q_R})$:
  $j \leftarrow 0$
  $\alpha \leftarrow \mathcal{B}'^{\mathcal{O},\mathcal{R}}(g, q, X)$
    If $\mathcal{B}'$ queries $\mathcal{R}$
      $j \leftarrow j + 1$
      Return $e_j$
  If $\alpha = (w^*, e^*, s^*, j^*)$
    Output $(j^*, w^*, e^*, s^*)$
  Else output $(0, \star, \star, \star)$
```

The forking algorithm F_A is the same as stated in the lemma, where σ is a tuple of the form $(w, e, s) \in \mathbb{G} \times \mathbb{Z}^2$. To make it clearer, we write it out in our context:

```
$F_A^{\mathcal{O}}(g, q, X)$:
  Pick coins $\rho$ for $A$ at random
  $e_1, \ldots, e_{q_R} \xleftarrow{\$} \mathbb{Z}$
  $(j^*, w^*, e^*, s^*) \leftarrow A(g, q, g^a, e_1, \ldots, e_{q_R}; \rho)$
  If $j^* = 0$
    Output $(0, \star, \star, \star, \star, \star, \star)$
  $\hat{e}_{j^*}, \ldots, \hat{e}_{q_R} \xleftarrow{\$} \mathbb{Z}$
  $(\hat{j}^*, \hat{w}^*, \hat{e}^*, \hat{s}^*) \leftarrow A(g, q, g^a, e_1, \ldots, e_{j^*-1}, \hat{e}_{j^*}, \ldots, \hat{e}_{q_R}; \rho)$
  If $j^* = \hat{j}^*$ and $e_{j^*} \neq \hat{e}_{\hat{j}^*}$
    Output $(1, w^*, e^*, s^*, \hat{w}^*, \hat{e}^*, \hat{s}^*)$
  Else output $(0, \star, \star, \star, \star, \star, \star)$
```

The algorithm $\mathcal{B}'$ is constructed upon F_A:

```
$\mathcal{B}^{\mathcal{O}}(g, q, X)$:
  If $(1, w^*, e^*, s^*, \hat{w}^*, \hat{e}^*, \hat{s}^*) \leftarrow F_A(g, q, X)$
    Compute $B_1^*, B_2^*, \hat{B}_1^*$ and $\hat{B}_2^*$
    Output $a \leftarrow \frac{B_1^* - \hat{B}_1^*}{\hat{B}_2^* - B_2^*} \bmod q$
  Else output $\perp$
```

If $X = g^a$, then we have $K_{j^*} = {y_{j^*}}^{B_1^* - aB_2^*}$ and $K_{j^*} = {y_{j^*}}^{\hat{B}_1^* - a\hat{B}_2^*}$. Since $e_{j^*} \neq \hat{e}_{j^*}$, we have $B_i^* \neq \hat{B}_i^*$ for $i \in \{1, 2\}$ by the fourth condition we required of the functions B_1 and B_2. Therefore, we can extract a by computing $(B_1^* - \hat{B}_1^*)/(\hat{B}_2^* - B_2^*) \bmod q$. Since A essentially outputs what $\mathcal{B}'$ outputs, it is clear that the accepting probability acc is equal to the success probability of $\mathcal{B}'$, $Adv_{\mathcal{B}'}^{\mathtt{GDL}'}$. Similarly, since $\mathcal{B}$ outputs a if and only F_A outputs $(1, w^*, e^*, s^*, \hat{w}^*, \hat{e}^*, \hat{s}^*)$, we have $Adv_{\mathcal{B}}^{\mathtt{GDL}} = \text{frk}$. Hence by the general forking lemma, we have

$$Adv_{\mathcal{B}}^{\mathtt{GDL}} \geq Adv_{\mathcal{B}'}^{\mathtt{GDL}'} (\frac{Adv_{\mathcal{B}'}^{\mathtt{GDL}'}}{q_R} - \frac{1}{q}) .$$

□

4.8 The Signature Schemes with $B_3(r, e, s) \neq 1$

We may expand the range of meta-ElGamal signature schemes to those which include $B_3(r, e, s)$. A meta-ElGamal signature scheme (B_1, B_2, B_3) with $B_3 \neq 1$ can be converted into a meta-ElGamal signature scheme $(B_1', B_2', 1)$ by setting

$$B_1'(r, e, s) = \frac{B_1(r, e, s)}{B_3(r, e, s)} \qquad \text{and} \qquad B_2'(r, e, s) = \frac{B_2(r, e, s)}{B_3(r, e, s)} .$$

Thus, an arbitrary meta-ElGamal signature scheme (B_1, B_2, B_3) can be transformed into a secure signcryption scheme as long as (B_1', B_2') satisfy the conditions listed in Section 4.3.

4.9 The Security of Meta-ElGamal Signatures

Despite having been proposed over fifteen years, no proof of security has ever been proposed for meta-ElGamal signature schemes. The techniques used to prove the unforgeability of the Zheng transform in Section 4.7 can be adopted to show the security of the meta-ElGamal signature scheme. However, we may take a short-cut given that we have already proven that the meta-ElGamal signcryption scheme obtained from the Zheng transform is unforgeable. It is well-known that an insider UF-CMA secure signcryption scheme can converted into a UF-CMA signature scheme by publishing a receiver's public/private key pair in the signature public key (see [15] for a proof). We introduce a formal definition for the derived signature scheme here:

Definition 4.9.1. *Consider a meta-ElGamal signature scheme SS parameterised by functions $(B_1, B_2, 1)$ and using a hash function H_2 as its encoding function. Let SC be the meta-ElGamal signcryption scheme obtained via the Zheng transform of SS. Then the* derived signature scheme *$dSS = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is defined as follows:*

```
$\mathcal{G}(1^k)$:
  $PP \xleftarrow{\$} Setup(1^k)$
  $(x_S, y_S) \xleftarrow{\$} KG_S(PP)$
  $(x_R, y_R) \xleftarrow{\$} KG_R(PP)$
  $sk \leftarrow (PP, x_S, y_S, y_R)$
  $pk \leftarrow (PP, y_S, x_R, y_R)$
  Output $(sk, pk)$

$\mathcal{S}(m, sk)$:
  $(PP, x_S, y_S, y_R) \leftarrow sk$
  $C \leftarrow SC(x_S, y_R, m)$
  Output $C$
$\mathcal{V}(m, C, pk)$:
  $(PP, y_S, x_R, y_R) \leftarrow pk$
  $m' \leftarrow USC(y_S, x_R, C)$
  If $m = m'$ then output $\top$
  Else output $\bot$
```

We can now relate the security of a meta-ElGamal signature scheme to the security of the corresponding derived signature scheme. The security notion we discuss here is UF-CMA security (Definition 2.3.4).

Theorem 4.9.2. *Let SC be a meta-ElGamal signcryption scheme with functions $(B_1, B_2, 1)$, and let SS be a meta-ElGamal signature scheme with the same functions $(B_1, B_2, 1)$. Then SS is UF-CMA secure if SC is UF-CMA secure.*

Proof: In order to prove this theorem, we are only required to show that *SS* is secure if *dSS* is secure. Let $\mathcal{A}'$ be an adversary that breaks the UF-CMA security of *SS*. We will construct another adversary $\mathcal{A}$ based on $\mathcal{A}'$ which would break the UF-CMA security of *dSS*. Consider the adversary shown in Figure 4.12.

```
$\mathcal{A}^{\mathcal{O}_S,H_1,H_2,H_3}(pk)$:                                  $\triangleright$ $\mathcal{O}_S$ is a signing oracle for $dSS$
  Parse $pk$ as $(PP, y_S, x_R, y_R)$
  $(m^*, (w^*, s^*)) \xleftarrow{\$} \mathcal{A}'^{\mathcal{O}'_S,H_S}(PP, y_S)$          $\triangleright$ $\mathcal{O}'_S$ is a simulated signing oracle
    If $\mathcal{A}'$ queries $\mathcal{O}'_S(m)$                               for $SS$
      Query $C \leftarrow \mathcal{O}_S(m)$
      Parse $C$ as $(c, \beta_1, \beta_2)$                            $\triangleright$ $H_S$ is the hash oracle for $SS$
      $w \leftarrow g^{\beta_1} y_S^{-\beta_2}$
      Query $r \leftarrow H_2(w)$
      Compute $(e, s)$ from $(r, \beta_1, \beta_2)$
      Return $(w, s)$
    If $\mathcal{A}'$ queries $H_S(m, w)$
      $K \leftarrow w^{x_R}$
      Query $e \leftarrow H_1(m\|y_S\|y_R\|K)$
      Return $e$
  $K^* \leftarrow w^{*x_R}$
  Query $e^* \leftarrow H_1(m^*\|y_S\|y_R\|K^*)$
  Query $r^* \leftarrow H_2(w^*)$
  $\beta_i^* \leftarrow B_i(r^*, e^*, s^*)$ for $i = 1, 2$
  $\tau^* \leftarrow H_3(y_S\|y_R\|K^*)$
  $c^* \leftarrow \mathcal{E}_{SE_{\tau^*}}(m^*)$
  Output $C^* \leftarrow (c^*, \beta_1^*, \beta_2^*)$
```

Figure 4.12: The adversary $\mathcal{A}$ against UF-CMA security of *dSS*

The action of the two oracles $\mathcal{O}'_S$ and H_S is consistent with the view that $\mathcal{A}'$ expects. (The same hash function H_2 is used by both $\mathcal{A}$ and $\mathcal{A}'$, so H_2's distribution is trivially correct.) Suppose $\mathcal{A}'$ outputs a valid forgery. Then $\mathcal{A}'$ cannot have queried its signing oracle $\mathcal{O}'_S$ on m and so $\mathcal{A}$ did not query its signing oracle on m. Moreover, if (w^*, s^*) is a valid signature for m^* in SS, then (r^*, e^*, s^*) satisfies

- $B_1(r^*, e^*, s^*) = x_S \cdot B_2(r^*, e^*, s^*) + t$ where $w^* = g^t$;
- $e^* = H_S(w^*, m^*) = H_1(m^* \| y_S \| y_R \| w^{*x_R})$; and
- $r^* = H_2(w^*)$.

Hence C^* is a valid signcryption ciphertext with underlying message m^* and so a valid *dSS* forgery on m^*. In other words, a successful adversary against the *SS* gives rise to a successful adversary against the *dSS* scheme (with the same success probability) and so *SS* must be secure if *SC* is unforgeable. □

4.10 Confidentiality of the GLZ Transform

In this section, we will prove the confidentiality of the GLZ transform. The proof is similar to the proof of the confidentiality of the Zheng transform.

Theorem 4.10.1. *If there exists a PPT adversary $\mathcal{A}$ against the IND-CCA2 property of the signcryption scheme, then there exist a PPT adversary $\mathcal{B}$ against the GDH problem and a PPT adversary $\mathcal{B}'$ against the IND-OT property of the symmetric encryption scheme such that*

$$Adv_{\mathcal{A}}^{\texttt{out-IND}}(k) \leq 2q_S \frac{q_1 + q_2 + 2q_S + 2q_U}{q} + 2\frac{q_S + q_U}{q} + 2Adv_{\mathcal{B}}^{\texttt{GDH}}(k) + Adv_{\mathcal{B}'}^{\texttt{PA}}(k) \ ,$$

where q_i is the maximum number of queries made by $\mathcal{A}$ to the H_i oracle, q_S is the maximum number of queries made to the signcryption oracle, and q_U is the maximum number of queries made to the unsigncryption oracle.

Proof: Let $\mathcal{A}$ be a PPT adversary against the IND-CCA2 property of the signcryption scheme. We will prove the theorem via a sequence of games [8, 34]. Each game will be parameterised by a bit b and we define W_i^b to be the event that $\mathcal{A}$ outputs 1 in the version of game G_i which uses bit b.

Game G_1:

G_1 is the game $\text{Expt}_{\mathcal{A}}^{\texttt{out-IND-b}}(k)$. Hence:

$$Adv_{\mathcal{A}}^{\texttt{out-IND}}(k) = |\Pr[W_1^1] - \Pr[W_1^0]| \ .$$

Game G_2:

G_2 simulates the random oracles, signcryption oracle and unsigncryption oracle internally using a series of lists. The lists L^{H_1}, L^{H_2}, L^{H_3} contain elements of the form $(m, y_S, y_R, w, K, e) \in \{0,1\}^* \times \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \mathbb{Z}_q$, $(w, r) \in \mathbb{G} \times \mathbb{Z}_q$, $(y_S, y_R, w, K, \tau) \in \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \{0,1\}^\ell$ corresponding to the input/output of the hash functions. Recall that $\mathcal{O}$ is a DDH oracle which determines whether (a, b, c) satisfies $\log_g b = \log_a c$.

We will change the way in which the oracles work, but first we must consider how to "address" elements of the lists L^{H_1} and L^{H_3}. As an example, we consider the list L^{H_1}. Elements of this list will always have the form $(m, y, y', \star, K, e)$ and $(m, y, y', w, \star, e)$. A query $H_1(m, y, y', w)$ should receive the answer e if either $(m, y, y', \star, K, e) \in L^{H_1}$ or if $(m, y, y', w, \star, e) \in L^{H_1}$, where $w = g^t$ and $K = y'^t$ for some t. We define the Def/Def' functions in Figure 4.13 to easily determine if an appropriate entry exists in the list.

This allow us to re-define the hash/signcryption/unsigncryption oracles as in Figure 4.14.

We assume that the challenge signcryption is computed as $\mathcal{O}_S(y_R^*, m_b)$. Since the hash functions are modeled as random oracles, and since we have worked hard to ensure consistency between the way in which elements in the

```
$Def_1(c, y, y', w)$:
  If $(c, y, y', w, \star, \cdot) = (c_i, y_i, y'_i, w_i, K_i, e_i)$ for $i \in I^{H_1}$
    $e \leftarrow e_i$
  Else if $(c, y, y', \cdot, \star, \cdot) = (c_i, y_i, y'_i, w_i, K_i, e_i)$ and $\mathcal{O}(w, y', K_i) = 1$ for $i \in I^{H_1}$
    $e \leftarrow e_i$
  Else
    $e \leftarrow \perp$
  Return $e$

$Def_2(w)$:
  If $(w, \cdot) = (w_i, r_i)$ for some $i \in I^{H_2}$
    $r \leftarrow r_i$
  Else
    $r \leftarrow \perp$
  Return $r$

$Def_3(y, y', K)$:
  If $(y, y', \star, K, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else if $(y, y', \cdot, \star, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ and $\mathcal{O}(y', w_i, K) = 1$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else
    $\tau \leftarrow \perp$
  Return $\tau$

$Def'_3(y, y', w)$:
  If $(y, y', w, \star, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else if $(y, y', \star, \cdot, \cdot) = (y_i, y'_i, w_i, K_i, \tau_i)$ and $\mathcal{O}(y', w, K_i) = 1$ for $i \in I^{H_3}$
    $\tau \leftarrow \tau_i$
  Else
    $\tau \leftarrow \perp$
  Return $\tau$
```

Figure 4.13: Functions which determine membership of the lists L^{H_1}, L^{H_2} and L^{H_3} from partial information

```
$H_1(c, y, y', w)$:
  $e \leftarrow Def_1(c, y, y', w)$
  If $e = \perp$
    $e \xleftarrow{\$} \mathbb{Z}_q$
    Add $(c, y, y', w, \star, e)$
      to $L^{H_1}$
  Return $e$
```

```
$H_2(w)$:
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$
      to $L^{H_2}$
  Return $r$
```

```
$H_3(y, y', K)$:
  $\tau \leftarrow Def_3(y, y', K)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y, y', \star, K, \tau)$
      to $L^{H_3}$
  Return $\tau$
```

```
$\mathcal{O}_S(y_R, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $\tau \leftarrow Def'_3(y_S^*, y_R, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S^*, y_R, w, \star, \tau)$ to $L^{H_3}$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow Def_1(c, y_S^*, y_R, w)$
  If $e = \perp$
    $e \xleftarrow{\$} \mathbb{Z}_q$
    Add $(y_S^*, y_R, w, \star, e)$ to $L^{H_1}$
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$ to $L^{H_2}$
  Solve $B_1(r, e, s) = x_S^* B_2(r, e, s) + t$
    for the variable $s$
  $C \leftarrow (c, B_1(r, e, s), B_2(r, e, s))$
  Return $C$
```

```
$\mathcal{O}_U(y_S, C)$:
  Parse $C$ as $(c, \beta_1, \beta_2)$
  $w \leftarrow g^{\beta_1} y_S^{-\beta_2}$
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$ to $L^{H_2}$
  Compute $(e, s)$ from $(\beta_1, \beta_2, r)$
  $e' \leftarrow Def_1(c, y_S, y_R^*, w)$
  If $e' = \perp$
    $e' \xleftarrow{\$} \mathbb{Z}_q$
    Add $(c, y_S, y_R^*, w, \star, e')$ to $L^{H_1}$
  If $e \neq e'$ then Return $\perp$
  Else
    $\tau \leftarrow Def'_3(y_S, y_R^*, w)$
    If $\tau = \perp$
      $\tau \xleftarrow{\$} \{0,1\}^\ell$
      Add $(y_S, y_R^*, w, \star, \tau)$ to $L^{H_3}$
    $m \leftarrow \mathcal{D}_{SE\tau}(c)$
    Return $m$
```

Figure 4.14: The hash/signcryption/unsigncryption oracles in Game G_2

```
$\mathcal{O}_S(y_R, m)$:
  $r, e, s \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^{B_1(r,e,s)} {y^*}_S^{-B_2(r,e,s)}$
  $\tau \leftarrow \mathit{Def}'_3(y^*_S, y_R, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y^*_S, y_R, w, \star, \tau)$ to $L^{H_3}$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e' \leftarrow \mathit{Def}_1(c, y^*_S, y_R, w)$
  If $e' \neq \perp$
    Output $\perp$ and halt all operations                    $\triangleright$We call this event $\perp_1$
  Add $(c, y^*_S, y_R, w, \star, e)$ to $L^{H_1}$
  $r' \leftarrow \mathit{Def}_2(w)$
  If $r' \neq \perp$
    Output $\perp$ and halt all operations                    $\triangleright$We call this event $\perp_2$
  Add $(w, r)$ to $L^{H_2}$
  $C \leftarrow (c, B_1(r,e,s), B_2(r,e,s))$
  Return $C$
```

Figure 4.15: The signcryption oracle $\mathcal{O}_S$ in Game G_3

list are “addressed”, we have that G_2 is equivalent to G_1. Hence:

$$\Pr[W_2^b] = \Pr[W_1^b] \ , \text{ where } b \in \{0,1\} \ .$$

Game G_3:

Note that in G_2, the receiver's private key x^*_R is not used at all. G_3 changes the action of the signcryption oracle so that it computes signcryption ciphertexts without using the sender's private key x^*_S. This is done by changing the signcryption oracle to act as shown in Figure 4.15.

The challenge ciphertext is computed as $\mathcal{O}_S(y^*_R, m_b)$. Since $r, e, s \xleftarrow{\$} \mathbb{Z}_q$, we have that w is uniformly distributed over $\mathbb{G}$ (by definition of $B_1(r,e,s)$ and $B_2(r,e,s)$). Thus, as long as $\perp_1$ and $\perp_2$ do not occur, we have that G_2 and G_3 are equivalent. Note that the size of L^{H_i} is bounded by $q_i + q_S + q_U$ for $i \in \{1,2\}$. Since w is distributed at random over $\mathbb{G}$, we have that the probability $\perp_1$ occurs in any execution of $\mathcal{O}_S$ is bounded by $(q_1 + q_S + q_U)/q$.

Similarly, the probability that $\perp_2$ occurs is bounded by $(q_2+q_S+q_U)/q$. Thus,

$$|\Pr[W_3^b] - \Pr[W_2^b]| \leq q_S \cdot \left(\frac{q_1 + q_2 + 2q_S + 2q_U}{q}\right) .$$

Game G_4:

G_4 is identical to G_3 except for a tiny change which ensures that the hash oracles only evaluate $H_3(y_R^*, y_S^*, K^*)$ during the computation of the challenge ciphertext. Formally, we can break this down into three cases:

1. The adversary could make a direct hash oracle query $H_3(y_R^*, y_S^*, K^*)$. This allows us to recover K^*. We define this to be event E_1. As we shall see, this leads to an algorithm which solves the GDH problem. Note that this event can be detected in polynomial-time using the DDH oracle.

2. The signcryption oracle $\mathcal{O}_S$ could attempt to make such a query; however, in that case, the value w computed by the signcryption oracle must be equal to w^*. Since both of these are computed at random, the probability that this occurs is bounded by q_S/q.

3. The unsigncryption oracle $\mathcal{O}_U$ could attempt to make such a query; however, in that case, the adversary must have submitted a query (c, B_1, B_2) such that $g^{B_1} y_S^{*B_2} = g^{B_1^*} y_S^{*B_2^*}$. This is split into two cases:

   - If $(B_1, B_2) = (B_1^*, B_2^*)$ then we must have $c \neq c^*$. This implies that $H_1(c, y_R^*, y_S^*, w^*) \neq e^*$ with probability $1 - 1/q$, where e^* is the hash value $H_1(c^*, y_R^*, y_S^*, w^*)$. In other words, the unsigncryption would output $\perp$ with probability $1 - 1/q$. We therefore alter the unsigncryption oracle so that it outputs $\perp$ if queried on (c, B_1^*, B_2^*). The probability that this is incorrect is bounded by q_U/q.

   - If $(B_1, B_2) \neq (B_1^*, B_2^*)$ then we must have $B_1^* \neq B_1$ and $B_2^* \neq B_2$. So we may recover t^* as $(B_1^* - B_1)/(B_2 - B_2^*)$ and recover K^* as ${y_R^*}^{t^*}$. We define this to be event E_2. As we shall see, this leads to

```
$\mathcal{B}^{\mathcal{O}}(g, X, Y)$:
  $w^* \leftarrow X$
  $y_R^* \leftarrow Y$
  $r^*, e^*, s^* \xleftarrow{\$} \mathbb{Z}_q$
  Add $(w^*, r^*)$ to $L^{H_2}$
  $\tau^* \leftarrow \{0,1\}^\ell$
  $y_S^* \leftarrow (g^{B_1(r^*,e^*,s^*)} w^{*-1})^{1/B_2(r^*,e^*,s^*)}$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1(y_S^*, y_R^*)$
  $c^* \leftarrow \mathcal{E}_{SE_{\tau^*}}(m_b)$
  $C^* \leftarrow (c^*, B_1^*(r^*, e^*, s^*), B_2^*(r^*, e^*, s^*))$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  If $E_1$ or $E_2$ occurred then output $K^*$
  Else output $\perp$
```

Figure 4.16: The algorithm $\mathcal{B}$ that solves the GDH problem

an algorithm which solves the GDH problem. Note that this event can be detected in polynomial-time by computing w.

We define an algorithm $\mathcal{B}$ which solves the GDH problem if E_1 or E_2 occurs in Figure 4.16.

If $\mathcal{A}$ makes any oracle queries, then they are answered using the simulations of H_1, H_2, H_3, $\mathcal{O}_S$ and $\mathcal{O}_U$ (with the exception that $\mathcal{O}_U$ returns $\perp$ on queries of the form (c, B_1^*, B_2^*)). Thus,

$$|\Pr[W_4^b] - \Pr[W_3^b]| \leq \frac{q_S + q_U}{q} + Adv_{\mathcal{B}}^{\mathtt{GDH}}(k)\,.$$

So, in G_4, the only time that $\mathcal{A}$ receives any data that depends on τ^* is when it receives C^* as the challenge ciphertext. Thus, there exists an adversary $\mathcal{B}' = (\mathcal{B}_1', \mathcal{B}_2')$ against IND-OT security by implicitly setting $H_3(y_S^*, y_R^*, K^*)$ to be equal to the hidden symmetric key used in the IND-OT game. The adversary $\mathcal{B}'$ is described in Figure 4.17 (we assume that all required variables are passed between $\mathcal{B}_1'$ and $\mathcal{B}_2'$ as part of the state variable).

$\mathcal{A}$'s oracle queries are answered using the simulators H_1, H_2, H_3, $\mathcal{O}_S$ and

```
$\mathcal{B}'_1(1^k)$:
  $x^*_S, x^*_R \xleftarrow{\$} \mathbb{Z}_q$
  $y^*_S \leftarrow g^{x^*_S}$; $y^*_R \leftarrow g^{x^*_R}$
  $r^*, e^*, s^* \xleftarrow{\$} \mathbb{Z}_q$
  $B^*_i \leftarrow B_i(r^*, e^*, s^*)$ for $i = 1, 2$
  $w^* \leftarrow g^{B^*_1} y^{*B^*_2}$
  Add $(w^*, r^*)$ to $L^{H_2}$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1(y^*_S, y^*_R)$
  Output $(m_0, m_1)$

$\mathcal{B}'_2(c^*)$:
  Add $(c^*, y^*_S, y^*_R, w^*, e^*)$ to $L^{H_1}$
  $C^* \leftarrow (c^*, B^*_1, B^*_2)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  If $\mathcal{A}$ made an "illegal" query
    then output 0
  Else output $b'$
```

Figure 4.17: The adversary $\mathcal{B}'$ against IND-OT security of the symmetric encryption scheme

$\mathcal{O}_U$. An "illegal" oracle query is a hash query $H_3(y^*_S, y^*_R, K^*)$ where $K^* = w^{*x^*_S}$, a signcryption query with $w = w^*$, or an unsigncryption oracle query with $w = w^*$ and $(B_1, B_2) \neq (B^*_1, B^*_2)$. An examination shows that

$$|\Pr[W^1_4] - \Pr[W^0_4]| = Adv^{\mathtt{PA}}_{\mathcal{B}'}(k)\,.$$

Thus, we can conclude

$$\begin{aligned} Adv^{\mathtt{out\text{-}IND}}_{\mathcal{A}}(k) &= |\Pr[W^1_1] - \Pr[W^0_1]| \\ &\leq 2q_S \frac{q_1 + q_2 + 2q_S + 2q_U}{q} + 2\frac{q_S + q_U}{q} + 2Adv^{\mathtt{GDH}}_{\mathcal{B}}(k) + Adv^{\mathtt{PA}}_{\mathcal{B}'}(k)\,. \end{aligned}$$

□

4.11 Unforgeability of the GLZ Transform

To prove the unforgeability of the GLZ transform, we could again apply the same proof technique used for proving the unforgeability of the Zheng transform. However, to avoid repetitive work, we choose to use the previous results to prove the unforgeablity of the GLZ transform. More precisely, we have proven that the signcryption schemes obtained from the Zheng transform are UF-CMA secure; this implies that the derived signature scheme is UF-CMA secure, and therefore the underlying meta-ElGamal signature scheme is UF-CMA secure. We are now going to prove that UF-CMA security of a meta-ElGamal signature scheme leads to UF-CMA security of the signcryption scheme resulting from applying the GLZ transform to that signature scheme.

Theorem 4.11.1. *Let SC be a signcryption scheme obtained from the GLZ transform using functions $B_1(r,e,s)$ and $B_2(r,e,s)$. If there exists a PPT adversary $\mathcal{A}$ against UF-CMA security of SC, then there exists a PPT adversary $\mathcal{A}'$ against UF-CMA security of the corresponding meta-ElGamal signature scheme for the same tuple of functions $(B_1(r,e,s), B_2(r,e,s), 1)$ such that:*

$$Adv_{\mathcal{A}}^{\texttt{in-UF}}(k) \leq \frac{q_3}{q} + (1 - q_S neg(k)) Adv_{\mathcal{A}'}^{\texttt{UF}}(k) \ .$$

where q_3 is the maximum number of queries made by $\mathcal{A}$ to the H_3 oracle, q_S is the maximum number of queries made to the signcryption oracle, and $neg(k)$ is some negligible function in k.

Proof: Let $\mathcal{A}$ be a PPT adversary against the UF-CMA property of the signcryption scheme. We will prove the theorem as before via a sequence of games [8, 34]. At the end of each game, $\mathcal{A}$ will output a tuple consisting of (pk_R^*, sk_R^*, C^*). On input the tuple, a checking algorithm will output 0 or 1 based on the three conditions listed in $\text{Expt}_{\mathcal{A}}^{\texttt{in-UF}}(k)$ in Definition 2.3.11. We define W_i to be the event that the checking algorithm outputs 1 in Game i.

Game G_1:

G_1 is the game $\text{Expt}_{\mathcal{A}}^{\texttt{in-UF}}(k)$. Hence:

$$Adv_{\mathcal{A}}^{\texttt{in-UF}}(k) = \Pr[W_1] \ .$$

Game G_2:

G_2 simulates the hash oracle H_3 using a series of lists. We adapt the notation introduced in the last proof:

```
$H_3(y, y', K)$:
  $\tau \leftarrow Def_3(y, y', K)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y, y', \star, K, \tau)$ to $L^{H_3}$
  Return $\tau$
```

This game is equivalent to G_1 and so we have $Pr[W_2] = Pr[W_1]$.

Game G_3:

G_3 simulates the signcryption oracle in the following way:

```
$\mathcal{O}_S(y_R, m)$:
  $\tau \xleftarrow{\$} \{0,1\}^\ell$
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $K \leftarrow y_R^t$
  If $\perp \leftarrow Def_3(y_S, y_R, K)$
    add $(y_S, y_R, w, K, \tau)$ to $L^{H_3}$
  Else
    Halt and output $\perp$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow H_1(c, y_S, y_R, w)$
  $r \leftarrow H_2(w)$
  Solve $B_1(r, e, s) = x_S B_2(r, e, s) + t$
    for the variable $s$
  $\beta_i \leftarrow B_i(r, e, s)$ for $i = 1, 2$
  $C \leftarrow (c, \beta_1, \beta_2)$
  Output $C$
```

The probability it outputs $\perp$ is bounded by q_3/q, since t is uniformly randomly chosen and w is uniformly distributed over $\mathbb{G}$. Thus,

$$|Pr[W_3] - Pr[W_2]| < \frac{q_3}{q}$$

Game G_4:

G_4 introduces two hash functions $\mathcal{E}, H_S$ and a signing oracle $\mathcal{O}^{sig}$ from the meta-ElGamal signature scheme with $(B_1, B_2, 1)$. The purpose of this is to construct a PPT algorithm $\mathcal{A}'$ based on $\mathcal{A}$ to break the UF-CMA security of the signature scheme. The queries made by $\mathcal{A}$ will be answered with the help of the signing oracle and the two hash oracles. The signing oracle is defined as:

```
$\mathcal{O}^{sig}(m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $r \leftarrow \mathcal{E}(w)$
  $e \leftarrow H_S(m, w)$
  Solve $B_1(r, e, s) = xB_2(r, e, s) + tB_3(r, e, s)$
    for the variable $s$
  Output $(w, s)$
```

We modify some oracles in G_3 as follows:

```
$H_1(c, y, y', w)$:
  $e \leftarrow H_S(c\|y\|y', w)$
  Add $(c, y, y', w, \star, e)$ to $L^{H_1}$
  Return $e$

$H_2(w)$:
  $r \leftarrow \mathcal{E}(w)$
  Add $(w, r)$ to $L^{H_2}$
  Return $r$
```

```
$\mathcal{O}_S(y_R, m)$:
  $\tau \xleftarrow{\$} \{0,1\}^\ell$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  Query $(w, s) \xleftarrow{\$} \mathcal{O}^{sig}(c\|y_S\|y_R)$
  If $\perp \leftarrow Def'_3(y_S, y_R, w)$
    add $(y_S, y_R, w, \star, \tau)$ to $L^{H_3}$
  Else
    halt and output $\perp$
  $r \leftarrow H_2(w)$
  $e \leftarrow H_1(c, y_S, y_R, w)$
  $\beta_i \leftarrow B_i(r, e, s)$ for $i = 1, 2$
  $C \leftarrow (c, \beta_1, \beta_2)$
  Output $C$
```

Note that the distribution of (w, s) generated by $\mathcal{O}^{sig}$ is exactly the same as that generated by $\mathcal{O}_S$ in G_3 by the definitions of these oracles. Under the random oracle model [7], the adversary cannot distinguish the difference between the oracles in G_3 and the oracles in G_4. Thus, we have:

$$Pr[W_4] = Pr[W_3].$$

Before we construct the algorithm $\mathcal{A}'$, let us recall the UF-CMA security game for signature schemes:

```
$\textsc{Expt}^{\mathtt{UF}}_{\mathcal{A}}(k)$:
  $(pk, sk) \leftarrow \mathcal{G}(1^k)$
  $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{sig}(sk,\cdot)}(pk)$
  Output 1 if
    (a) $\mathcal{V}(pk, m^*, \sigma^*) = \top$
    (b) $\mathcal{A}$ never queried $\mathcal{O}^{sig}(sk, m^*)$
  Else output 0,
```

where $\mathcal{O}^{sig}$ is a signing oracle.

Suppose we have $(y_S^*, x_S^*) \leftarrow \mathcal{G}(1^k)$ and $\mathcal{A}'$ is given y_S^*. Then the adversary $\mathcal{A}'$ that would break the UF-CMA security of the signature scheme can be described as follows:

```
$\mathcal{A}'^{\mathcal{O}^{sig}}(y_S^*)$:
  $(pk_R^*, sk_R^*, C^*) \xleftarrow{\$} \mathcal{A}(y_S^*)$
  Parse $C^*$ as $(c^*, \beta_1^*, \beta_2^*)$
  $w^* \leftarrow g^{\beta_1^*} y_S^{*\,-\beta_2^*}$
  $r^* \leftarrow \mathcal{E}(w^*)$
  Compute $(e^*, s^*)$ from $\beta_1^*, \beta_2^*$ and $r^*$
  Output $(c^* \| y_S^* \| y_R^*, w^*, s^*)$
```

If C^* is a valid forgery for the signcryption scheme, then we have

$$H_1(c^*, y_S^*, y_R^*, w^*) = e^*, \text{ where } e^* \text{ is computed from } \beta_1, \beta_2 \text{ and } r^*.$$

This implies $H_S(c^* \| y_S^* \| y_R^*, w^*) = e^*$ and we have $\top \leftarrow \mathcal{V}(c^* \| y_S^* \| y_R^*, w^*, s^*)$, where $\mathcal{V}$ is the verification algorithm in the signature scheme. In other words, if C^* is a valid forgery then so is $(c^* \| y_S^* \| y_R^*, w^*, s^*)$. Note that if $\mathcal{A}$ breaks the UF-CMA security, then $\mathcal{A}$ has never queried $\mathcal{O}_S(y_R^*, m^*)$. However, it is possible that $\mathcal{A}'$ has queried $c^* \| y_S^* \| y_R^*$ on $\mathcal{O}^{sig}$. This happens only if the following conditions are met:

1. $\mathcal{A}$ submits a signcryption query on some message $m \neq m^*$ and the receiver's public key y_R^*,

2. the signcryption oracle picks τ randomly, and it turns out that $\mathcal{E}_{SE}(m, \tau)$ is c^*.

We want to find the probability of this event, which we call E. For the symmetric encryption scheme, we required in Section 2.3.1 that given a message m and a ciphertext c , the probability that the randomly chosen key τ satisfies $\mathcal{E}_{SE}(m,\tau) = c$ is negligible in the security parameter. We denote this negligible probability by $\text{neg}(k)$. Note that this probability is for each execution of the signcryption oracle. Therefore $Pr[E] = q_S\text{neg}(k)$. Thus, we can bound $Pr[W_4]$ by $(1 - q_S\text{neg}(k))Adv^{\texttt{UF}}_{\mathcal{A}'}(k)$. To conclude, we have

- $Adv^{\texttt{in-UF}}_{\mathcal{A}}(k) = Pr[W_1]$,
- $Pr[W_2] = Pr[W_1]$,
- $|Pr[W_3] - Pr[W_2]| \le \frac{q_3}{q}$,
- $Pr[W_4] = Pr[W_3]$,
- $Pr[W_4] \le (1 - q_S\text{neg}(k))Adv^{\texttt{UF}}_{\mathcal{A}'}(k)$.

Combining all of these, we get

$$Adv^{\texttt{in-UF}}_{\mathcal{A}}(k) \le \frac{q_3}{q} + (1 - q_S\text{neg}(k))Adv^{\texttt{UF}}_{\mathcal{A}'}(k) \ .$$

□

4.12 Summary

In this chapter, we provided two transforms from a meta-ElGamal signature scheme to a signcryption scheme: the Zheng transform and the GLZ transform, inspired by the Zheng signcryption scheme [37] and the Gamage *et al.* signcryption scheme [18] respectively. Both transforms produces signcryption schemes that are outsider confidential and insider unforgeable. Moreover, the GLZ transform offers public verifiability, which allows anyone who has the knowledge of receiver's public key to verify the ciphertext. A large class of signcryption schemes can be constructed in this way by choosing different functions, B_1 and B_2. In this work, we have restricted ourselves to those functions of the form $(-1)^{a_0} r^{a_1} e^{a_2} s^{a_3}$ for $a_0 \in \{0, 1\}, a_1, a_2, a_3 \in \{-1, 0, 1\}\}$, and achieved provable security for the resulting signcryption schemes. It would be interesting to look into functions of other forms, and explore new signcryption schemes.

Chapter 5

From Outsider to Insider

In Chapter 4, we proved that the Zheng transform results in outsider IND-CCA2 secure signcryption schemes. It is quite natural to look for improvements to achieve insider confidentiality for the transform. In this chapter, we propose a modification to the Zheng transform so that the resulting signcryption schemes are insider secure. This result offers the users a choice between insider security and outsider security. Moreover, the cost to have this choice is merely an extra exponentiation operation.

5.1 The Modified Zheng Transform

Like the Zheng transform, the modified Zheng transform turns a meta-ElGamal signature scheme into a signcryption scheme. Without loss of generality, we assume the meta-ElGamal signature schemes are all parameterised by a triplet $(B_1(r,e,s), B_2(r,e,s), 1)$ which satisfies the list of properties in Section 4.3. For the modified Zheng transform, the setup algorithm and the key generation algorithms are the same as those for the Zheng transform.

The algorithm *Setup* generates a group $\mathbb{G}$ with a generator g whose order is q, where q is a k-bit prime. The sender key generation algorithm KG_S chooses a private key $x_S \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes the public key as $y_S \leftarrow g^{x_S}$. The receiver key generation algorithm KG_R chooses a private key $x_R \overset{\$}{\leftarrow} \mathbb{Z}_q$ and computes the public key as $y_R \leftarrow g^{x_R}$.

```
$SC(x_S, y_R, m)$:
  $t \xleftarrow{\$} \mathbb{Z}_q$
  $w \leftarrow g^t$
  $K \leftarrow {y_R}^t$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $c \leftarrow \mathcal{E}_{SE}(m, \tau)$
  $e \leftarrow H_1(m, y_S, y_R, K)$
  $r \leftarrow H_2(w)$
  Solve $B_1(r, e, s) = x_S B_2(r, e, s) + t$
    for the variable $s$
  $\beta_i \leftarrow B_i(r, e, s)$ for $i = 1, 2$
  $z \leftarrow g^{\beta_1}$
  $C \leftarrow (c, z, \beta_2)$
  Output $C$
```

```
$USC(y_S, x_R, C)$
  Parse $C$ as $(c, z, \beta_2)$
  $w \leftarrow z{y_S}^{-\beta_2}$
  $K \leftarrow w^{x_R}$
  $\tau \leftarrow H_3(y_S, y_R, K)$
  $m \leftarrow \mathcal{D}_{SE}(c, \tau)$
  $e \leftarrow H_1(m, y_S, y_R, K)$
  $r \leftarrow H_2(w)$
  Compute $s'$ from $e, r, \beta_2$
  $\beta_1' \leftarrow B_1(r, e, s')$
  If $g^{\beta_1'} \neq z$ then output $\perp$
  Else output $m$
```

Figure 5.1: The modified Zheng transform

The signcryption and unsigncryption algorithms make use of an IND-OT secure symmetric encryption scheme $(\mathcal{E}_{SE}, \mathcal{D}_{SE})$ with keyspace $\{0,1\}^\ell$, where ℓ is the security parameter for the symmetric encryption scheme, and a set of hash functions:

$$H_1 : \{0,1\}^* \times \mathbb{G}^3 \to \mathbb{Z}_q \,,\; H_2 : \mathbb{G} \to \mathbb{Z}_q \,, \text{ and } H_3 : \mathbb{G}^3 \to \{0,1\}^\ell \,.$$

The description of the signcryption and unsigncryption algorithms are given in Figure 5.1.

The modification is essentially a small change in the ciphertext: instead of outputting C as (c, β_1, β_2), it outputs C as $(c, g^{\beta_1}, \beta_2)$. Correspondingly, the unsigncryption algorithm checks if $g^{\beta_1'} = g^{\beta_1}$ instead of $e' = e$. We claim that this modification helps the scheme to achieve insider confidentiality.

5.2 The Security Proofs

In this section, we will provide two security proofs. Referring to the insider security models for signcryption given in Definition 2.3.9 and Definition 2.3.11, we will prove our signcryption schemes resulting from the modified Zheng transform are insider IND-CCA2 secure as well as insider UF-CMA secure.

5.2.1 Insider Confidentiality

Theorem 5.2.1. *Let $(Setup, KG_S, KG_R, SC, USC)$ be a signcryption scheme defined by the modified Zheng transform and functions $(B_1, B_2, 1)$. Suppose there exists a PPT adversary $\mathcal{A}$ against the IND-CCA2 property of the signcryption scheme in the insider model. Then there exist a PPT adversary $\mathcal{B}$ against the GDH problem and a PPT adversary $\mathcal{B}'$ against the IND-OT property of the symmetric encryption scheme such that:*

$$Adv_{\mathcal{A}}^{\mathtt{IND}}(k) \leq \frac{2q_1 + 2q_3 + 4q_U}{q} + 2Adv_{\mathcal{B}}^{\mathtt{GDH}}(k) + Adv_{\mathcal{B}'}^{\mathtt{OT}}(k)\,,$$

where q_i is the maximum number of queries made by $\mathcal{A}$ to the H_i oracle for $i = 1, 3$ and q_U is the maximum number of queries made to the unsigncryption oracle.

Proof: We are going to adapt the notion introduced in the last chapter and prove the theorem via a series of games [8, 34]. As before, each game will be parameterised by a bit b. We define W_i^b be the event that $\mathcal{A}$ outputs 1 in the version of game G_i which uses bit b, where $\mathcal{A}$ is a PPT adversary against the IND-CCA2 property of the signcryption scheme in the insider model.

Before we start game hopping, we give an overview of the proof.

1. The first game will be the same as $\textsc{Expt}_{\mathcal{A}}^{\mathtt{in\text{-}IND\text{-}b}}(k)$ defined in Definition 2.3.9.

2. The second game will simulate the unsigncryption oracle so that the receiver's private key is not used.

3. The third game will have a new rule so that we can simulate the signcryption algorithm without the sender's private key in the fourth game.

4. The fourth game will relate the security of the signcryption scheme to the advantage of solving the GDH problem.

5. The fifth game will relate the security of the signcryption scheme to the IND-OT property of the symmetric encryption scheme.

The details of the games are described below:

Game G_1:

G_1 is the game $\textsc{Expt}_{\mathcal{A}}^{\texttt{in-IND-b}}(k)$ described in Definition 2.3.9. Hence,

$$Adv_{\mathcal{A}}^{\texttt{in-IND}}(k) = |\Pr[W_1^1] - \Pr[W_1^0]| \ .$$

Game G_2:

G_2 simulates the random oracles and the unsigncryption oracle internally using three lists, L^{H_1}, L^{H_2} and L^{H_3}, corresponding to the input/output of the hash functions, where

1. L^{H_1} contains elements of the form

$$(m, y_S, y_R, w, K, e) \in \{0,1\}^* \times \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \mathbb{Z}_q \ ;$$

2. L^{H_2} contains elements of the form

$$(w, r) \in \mathbb{G} \times \mathbb{Z}_q \ ; \ \text{and}$$

3. L^{H_3} contains elements of the form

$$(y_S, y_R, w, K, \tau) \in \mathbb{G}^2 \times \overline{\mathbb{G}}^2 \times \{0,1\}^\ell \ .$$

We use the Def/Def' functions defined in Figure 4.5 to easily determine if an appropriate entry exists in the list. The Def functions are used when a proper hash query is submitted. That is, the hash queries include the exact input for the corresponding hash functions. The Def' functions are used when some part of the input of the corresponding hash function is unknown. Both

```
$H_1(m, y, y', K)$:
  $e \leftarrow Def_1(m, y, y', K)$
  If $e = \perp$
    $e \xleftarrow{\$} \mathbb{Z}_q$
    Add $(m, y, y', \star, K, e)$ to $L^{H_1}$
  Return $e$

$H_2(w)$:
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$
      to $L^{H_2}$
  Return $r$

$H_3(y, y', K)$:
  $\tau \leftarrow Def_3(y, y', K)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y, y', \star, K, \tau)$ to $L^{H_3}$
  Return $\tau$
```

```
$\mathcal{O}_U(y_S, C)$:
  Parse $C$ as $(c, z, \beta_2)$
  $w \leftarrow z{y_S}^{-\beta_2}$
  $r \leftarrow Def_2(w)$
  If $r = \perp$
    $r \xleftarrow{\$} \mathbb{Z}_q$
    Add $(w, r)$ to $L^{H_2}$
  $\tau \leftarrow Def'_3(y_S, y_R^*, w)$
  If $\tau = \perp$
    $\tau \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S, y_R^*, w, \star, \tau)$ to $L^{H_3}$
  $m \leftarrow \mathcal{D}_{SE}(c, \tau)$
  $e' \leftarrow Def'_1(m, y_S, y_R^*, w)$
  If $e' = \perp$
    $e' \xleftarrow{\$} \mathbb{Z}_q$
    Add $(m, y_S, y_R^*, w, \star, e',)$ to $L^{H_1}$
  Compute $s'$ from $(r, e', \beta_2)$
  $\beta'_1 \leftarrow B_1(r, e', s')$
  $z' \leftarrow g^{\beta'_1}$
  If $z' \neq z$ then return $\perp$
  Else return $m$
```

Figure 5.2: The hash/unsigncryption oracles in Game G_2

types of functions record the input and output of the hash functions. They may access the DDH oracle to check if an entry is on the list. This now allows us to re-define the hash/unsigncryption oracles as in Figure 5.2.

Note that the receiver's private key x_R^* is not used during the unsigncryption process. We assume that the challenge ciphertext is computed as $SC(x_S^*, y_R, m_b)$ using re-defined hash oracles. Since the hash functions are modeled as random oracles, and since the lists ensure consistency amongst the queries, we have that G_2 is identical to G_1. In other words,

$$\Pr[W_2^b] = \Pr[W_1^b] \quad \text{for } b \in \{0, 1\} \ .$$

Game G_3:

G_3 will have a new rule. Before we explain the new rule, we first define a particular type of query. We know that $\mathcal{A}_1$ will output $(m_0, m_1, x_S^*, y_S^*, \omega)$ and the challenger will apply the signcryption algorithm to the input (x_S^*, y_S^*, m_b) and get $C^* = (c^*, z^*, \beta_2^*)$. We have $w^* = z^* {y_S^*}^{\beta_2^*}$. We define a w^* query to be any one of the following three queries:

- a hash query $H_1(m, y_S^*, y_R^*, K^*)$, where (K^*, w^*, y_R^*) is a Diffie-Hellman triplet;
- a hash query $H_3(y_S^*, y_R^*, K^*)$, where (K^*, w^*, y_R^*) is a Diffie-Hellman triplet;
- an unsigncryption query (c, z, β_2) with sender's public key y_S^* such that $z {y_S^*}^{\beta_2} = z^* {y_S^*}^{\beta_2^*} = w^*$.

Note that one cannot recognise a w^* query before the challenge ciphertext is produced. The new rule is that if the adversary makes any w^* query before the challenge ciphertext is produced, then the adversary loses the game. That is, the adversary can make any queries and, once the challenge ciphertext is produced, the challenger will check if any previous query is a w^* query. Note that this can be done with the help of a DDH oracle. The adversary will lose the game if there is one. We shall analyse the probability of the adversary losing the game in this way.

Since w^* is randomly independently generated during the signcryption process, we have that a particular query (before the challenge ciphertext is generated) happens to be a w^* query with probability $1/q$. Hence the probability of the adversary losing the game in this way is bounded by $(q_1 + q_2 + q_U)/q$. Hence we can claim that the adversary will not notice the difference between G_3 and G_2 because the probability of the adversary losing the game in this way is negligible. More precisely,

$$|\Pr[W_3^b] - \Pr[W_2^b]| \leq \frac{q_1 + q_2 + q_U}{q}.$$

Game G_4:

G_4 changes the way of the computing challenge ciphertext so that the sender's private key x_S^* is not used. This is done by changing the signcryption algorithm as shown in Figure 5.3.

As in Figure 5.3, we have changed the way that the challenge ciphertext is produced. We need to show that the distribution of the ciphertext (c^*, z^*, β_2^*) produced by the simulation and by the real signcryption algorithm are indistinguishable to the adversary. As the generation of c^* in the simulation is the same as in the real signcryption algorithm, c^* has the same distribution in G_4 and in G_3. The rest is done by proving the following two claims:

Claim 1: The distribution of β_2^* in G_4 and in G_3 are indistinguishable.

Proof of Claim 1: In G_4, β_2^* is uniformly distributed over $\mathbb{Z}_q$. In G_3, β_2^* is computed as $B_2(r^*, e^*, s^*)$. By our assumption on B_2, we have that the distribution of β_2^* is indistinguishable from the uniform distribution in G_3. Hence, β_2^* in G_4 and in G_3 have indistinguishable distribution. $\square$

Claim 2: The distribution of z^* in G_4 and in G_3 are indistinguishable.

```
$SC\text{-}Challenge(y_S^*, y_R^*, m_b)$:
  $w^* \xleftarrow{\$} \mathbb{G}$
  $r^* \leftarrow H_2(w)$
  $\tau^* \leftarrow Def'_3(y_S^*, y_R^*, w^*)$
  If $\tau^* = \perp$
    $\tau^* \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S^*, y_R^*, w^*, \star, \tau^*)$ to $L^{H_3}$
  $c^* \leftarrow \mathcal{E}_{SE}(m_b, \tau^*)$
  $\beta_2^* \xleftarrow{\$} \mathbb{Z}_q$
  $z^* \leftarrow (y_S^*)^{\beta_2^*}(w^*)$
  $e' \leftarrow Def'_1(m_b, y_S^*, y_R^*, w^*)$
  If $e' \neq \perp$
    Output $\perp$ and halt all operations                    $\triangleright$We call this event $\perp_1$
  Add $(m_b, y_S^*, y_R^*, w^*, \star, \star)$ to $L^{H_1}$
  $C^* \leftarrow (c^*, z^*, \beta_2^*)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  Output $C^*$
```

Figure 5.3: The signcryption algorithm for generating the challenge ciphertext in G_4

Proof of Claim 2: In G_4, z^* is computed as ${y_S^*}^{\beta_2^*}w^*$, where y_S^* is fixed, β_2^* is uniformly random over $\mathbb{Z}_q$, and w^* is uniformly random over $\mathbb{G}$. Without loss of generality, assuming y_S^* is not the identity in $\mathbb{G}$, y_S^* is a generator of $\mathbb{G}$ since $\mathbb{G}$ is of prime order. This implies ${y_S^*}^{\beta_2^*}$ is a random element in $\mathbb{G}$. Hence z^* is the product of two random elements in $\mathbb{G}$. Therefore, z^* has a uniform distribution over $\mathbb{G}$ in G_4. In G_3, z^* is computed as $g^{\beta_1^*}$, where $\beta_1^* = B_1(r^*, e^*, s^*)$. By our assumption on B_1, β_1 has a distribution that is indistinguishable from a uniform distribution over $\mathbb{Z}$. This implies z^* has a distribution that is indistinguishable from a uniform distribution over $\mathbb{G}$ in G_3. Therefore, the distribution of z^* in G_4 and in G_3 are indistinguishable. $\square$

We have shown that the distribution of the challenge ciphertext in G_4 is unchanged from the adversary's point of view. The next problem with the simulation is the event $\perp_1$. It seems that the adversary would notice the difference caused by the simulation if $\perp_1$ happened. However, $\perp_1$ implies the hash value of $H_1(m_b, y_S^*, y_R^*, K^*)$ has been defined before the computation of the challenge ciphertext. In other words, the event $\perp_1$ happens only if there is a w^* query. In G_3, the adversary loses the game if there is a w^* query. Therefore, according to the new rule in G_3, if $\perp_1$ happens, the adversary will lose the game. As the occurrence of $\perp_1$ leads to the loss of the game, the adversary will not notice this difference between G_4 and G_3. From now on, we may assume $\perp_1$ does not occur.

The last problem with the simulation is more complicated. The simulation generated w^* and β_2^*, and computed z^* as ${y_S^*}^{\beta_2^*}w^*$. This implies β_1^* is fixed by the equation $g^{\beta_1^*} = z^*$. We have now fixed values of $(r^*, \beta_1^*, \beta_2^*)$ and hence, e^* is fixed according to the property listed in Section 4.3. We then let the hash value of $H_1(m_b, y_S^*, y_R^*, K^*)$ be e^*. By doing this, the simulation can create a valid challenge ciphertext. However, in the simulation, we are unable to work

```
$\mathcal{B}^{\mathcal{O}}(g, X, Y)$:
  $w^* \leftarrow X$
  $y_R^* \leftarrow Y$
  $(m_0, m_1, x_S^*, y_S^*, \omega) \xleftarrow{\$} \mathcal{A}_1(y_R^*)$
  $\tau^* \leftarrow Def'_3(y_S^*, y_R^*, w^*)$
  If $\tau^* = \perp$
    $\tau^* \xleftarrow{\$} \{0,1\}^\ell$
    Add $(y_S^*, y_R^*, w^*, \star, \tau^*)$ to $L^{H_3}$
  $c^* \leftarrow \mathcal{E}_{SE}(m_b, \tau^*)$
  $r^* \leftarrow H_2(w)$
  $\beta_2^* \xleftarrow{\$} \mathbb{Z}_q$
  $z^* \leftarrow (y_S^*)^{\beta_2^*}(w^*)$
  $C^* \leftarrow (c^*, z^*, \beta_2^*)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  If $E_1$ occurred then output $K^*$
  Else output $\perp$
```

Figure 5.4: The algorithm solving GDH problem with $\mathcal{A}$ as a sub-routine given E_1 occurs

out β_1^* from z^*, and therefore we cannot workout e^*, yet both β_1^* and e^* are implicitly defined. This would be a problem if the adversary makes an H_1 query on input (m_b, y_S^*, y_R^*, K^*) or the adversary triggers the unsigncryption oracle to make such a query later. We deal with these two cases separately.

Let E_1 denote the event that the adversary makes an H_1 query on input (m_b, y_S^*, y_R^*, K^*) directly. We are going to show that the event E_1 leads to an algorithm which solves the GDH problem.

Let $\mathcal{B}$ be an adversary against the GDH problem. In particular, $\mathcal{B}$ has access to a DDH oracle $\mathcal{O}$. We define $\mathcal{B}$ as in Figure 5.4.

As shown in Figure 5.4, $\mathcal{B}$ uses $\mathcal{A}$ as a sub-routine and solves the GDH problem if E_1 happens. This implies:

$$Pr[E_1] \leq Adv_{\mathcal{B}}^{\texttt{GDH}}(k)\,.$$

As we have assumed the hardness of the GDH problem, $Adv_{\mathcal{B}}^{\texttt{GDH}}(k)$ is negligible in k. Hence $Pr(E_1)$ is negligible in k.

Now let E_2 denote the event that the adversary triggers the unsigncryption oracle to make an H_1 query on input (m_b, y_S^*, y_R^*, K^*). To overcome this problem, we make the unsigncryption oracle output $\perp$ if this happens. Let $\perp_2$ denote the event that the unsigncryption oracle outputs $\perp$ wrongly. For a better presentation, we will analyse this event in the next game.

To conclude, we have:

$$|\Pr[W_4^b] - \Pr[W_3^b]| \leq Pr[E_1] + Pr[E_2] \leq Adv_{\mathcal{B}}^{\texttt{GDH}}(k) + \Pr[\perp_2]\,.$$

Game G_5:

G_5 deals with the case that the adversary triggers the unsigncryption oracle to make hash oracle query $H_1(m_b, y_S^*, y_R^*, K^*)$ after the challenge ciphertext is produced. We make the unsigncryption oracle output $\perp$ if this happens in G_4 and we are going to show that this change cannot be noticed by the adversary in this game. Note that we are not going to make any change in G_5. The purpose of G_5 is to determine some upper bounds for $\Pr[\perp_2]$ and $Pr[W_4^b]$.

In order to trigger the unsigncryption oracle to make such queries, the adversary must have submitted a query (c, z, β_2) with sender's public key y_S^* such that $zy_S^{*\,-\beta_2} = z^*y_S^{*\,-\beta_2^*} = w^*$, i.e., a w^* query. Note that we assume E_1 does not occur in this case. We analyse this case by considering a series of games.

We define $G_{5\text{-}i}$ to be the game that the unsigncryption oracle output $\perp$ to a w^* query for the first i-th unsigncryption queries after the challenge ciphertext is produced for $i = 1, 2, \ldots, q_U$. So $G_{5\text{-}1}$ is the game that is exactly the same as before except that the unsigncryption oracle will output $\perp$ if the first unsigncryption query after the challenge ciphertext is produced is a w^* query. Game $G_{5\text{-}2}$ is exactly the same as $G_{5\text{-}1}$ except that the unsigncryption oracle will output $\perp$ if the second unsigncryption query after the challenge ciphertext is produced is a w^* query. We can extend this definition to $i = 0$, where the game $G_{5\text{-}0}$ is exactly the same as G_3.

Now, suppose we are in $G_{5\text{-}i}$ and the i-th unsigncryption query after the challenge ciphertext is produced is a w^* query. Let us analyse the probability that the unsigncryption oracle outputs $\perp$ wrongly. A w^* query is of two possible forms: (c, z^*, β_2^*) and (c, z, β_2) for some $z \neq z^*$, some $\beta_2 \neq \beta_2^*$ and some $c \neq c^*$. Note that one may have queries also of the form (c^*, z, β_2), (c^*, z, β_2^*), (c^*, z^*, β_2), (c, z^*, β_2), or (c, z, β_2^*). As $w^* = z^* {y_S^*}^{\beta_2^*}$, it is clear that $z {y_S^*}^{\beta_2^*} \neq w^*$ and $z^* {y_S^*}^{\beta_2} \neq w^*$ for $z \neq z^*$ and $\beta_2 \neq \beta_2^*$, so the last four forms cannot give rise to a w^* query. If (c^*, z, β_2) is queried and suppose $z {y_S^*}^{\beta_2} = w^*$, then we have $Def'_3(y_S^*, y_R^*, w^*) = \tau^*$. This implies the underlying message of c^* is m_b and $H_1(m_b, y_S^*, y_R^*, K^*) = e^*$. (Although we do not know the value of e^*, it is implicitly defined and we denote it by e^*.) We also have $H_2(w^*) = r^*$. Solving the equation $B_1(r^*, e^*, s) = x_S^* B_2(r^*, e^*, s) + t^*$ gives $s = s^*$. By the first property listed in Section 4.3, $s = s^*$ is the only solution. We get $\beta_2 = B_2(r^*, e^*, s^*) = \beta_2^*$. This is a contradiction to the condition $\beta_2 \neq \beta_2^*$. Hence, (c^*, z, β_2) cannot give rise to a w^* query.

As discussed above, we only need to consider two possible forms of a w^* query: (c, z^*, β_2^*) and (c, z, β_2). Let m denote the underlying message of the ciphertext c and $r^* \leftarrow H_2(w^*)$.

1. Suppose (c, z^*, β_2^*) is queried. Then $H_1(m, y_S^*, y_R^*, K^*)$ has never been queried because of our assumption that E_1 does not occur. (Note that there exists only one encryption of m_b by our assumption about the symmetric encryption scheme, hence $m \neq m_b$.) This implies the hash value $H_1(m, y_S^*, y_R^*, K^*)$ is randomly chosen when the unsigncryption oracle is trying to respond to the query and it happens to be e^* with probability $1/q$. The unsigncryption oracle then work out s from (β_2^*, r^*, e) and calculate $\beta_1 = B_1(r^*, e, s)$. We know that $B_2(r^*, e, s) = \beta_2^* = B_2(r^*, e^*, s^*)$. Suppose $\beta_1 = \beta_1^*$. Then we have $e = e^*$ by the property of B_1 and

B_2. However, the probability that $e = e^*$ is $1/q$. In other words, the probability that $\beta_1 \neq \beta_1^*$ is $1 - 1/q$. This implies the probability that $g^{\beta_1} \neq g^{\beta_1^*} = z^*$ is $1 - 1/q$. Hence, the unsigncryption oracle will output $\perp$ correctly with probability $1 - 1/q$. Equivalently, the probability that the unsigncryption will output $\perp$ wrongly to this particular query is $1/q$.

2. Suppose (c, z, β_2) is queried. Let β_1 be such that $g^{\beta_1} = z$. From the third property on B_1, B_2 in Section 4.3, it follows that β_1 and β_2 determine the hash value $e = H_1(m, y_S^*, y_R^*, K^*)$. Since we assume that the event E_1 does not occur, the unsigncryption oracle will pick a random value e' for $H_1(m, y_S^*, y_R^*, K^*)$ when trying to respond to the query. The probability that $e' = e$ is $1/q$. As argued in the above case, the probability that the unsigncryption oracle will output $\perp$ wrongly is the same as the probability that $e' = e$, i.e., $1/q$.

Thus, we have:

$$\left|\Pr[W^b_{5\text{-}(i+1)}] - \Pr[W^b_{5\text{-}i}]\right| = \frac{1}{q} \quad \text{for } i = 0, 1, \ldots, q_U \, .$$

Hence

$$\begin{aligned} \left|\Pr[W^b_{5\text{-}0}] - \Pr[W^b_{5\text{-}q_U}]\right| &\leq \sum_{i=1}^{q_U-1} \left|\Pr[W^b_{5\text{-}(i+1)}] - \Pr[W^b_{5\text{-}i}]\right| \\ &= \frac{q_U}{q} \, . \end{aligned}$$

To summarise, we made a change to the unsigncryption oracle so that it returns $\perp$ to any w^* query. The probability that this is wrong is bounded by q_U/q. That is, $Pr(\perp_2)$ is bounded by q_U/q. To simplify the notation, we define G_5 to be $G_{5\text{-}q_U}$. Note that G_5 is exactly the same as G_4. Thus,

$$|\Pr[W_4^b] - \Pr[W_3^b]| \leq Adv_{\mathcal{B}}^{\texttt{GDH}}(k) + Pr[\perp_2] \leq Adv_{\mathcal{B}}^{\texttt{GDH}}(k) + \frac{q_U}{q} \, .$$

Now, in G_5, the only time that $\mathcal{A}$ receives any data that depends on τ^* is when it receives c^* as part of the challenge ciphertext. Thus, we can give an adversary $\mathcal{B}' = (\mathcal{B}'_1, \mathcal{B}'_2)$ against the IND-OT security by implicitly setting $H_3(y_S^*, y_R^*, K^*)$ to be equal to the hidden symmetric key used in the IND-OT game. The adversary $\mathcal{B}'$ runs as follows (we assume that all required variables are passed between $\mathcal{B}'_1$ and $\mathcal{B}'_2$ as part of the state variable):

```
$\mathcal{B}'_1(1^k)$:
  $x_R^* \xleftarrow{\$} \mathbb{Z}_q$
  $y_R^* \leftarrow g^{x_R^*}$
  $e^*, s^* \xleftarrow{\$} \mathbb{Z}_q$
  $z^* = g^{e^*}$
  $(m_0, m_1, x_S^*, y_S^*, \omega) \xleftarrow{\$} \mathcal{A}_1(y_R^*)$
  Output $(m_0, m_1)$

$\mathcal{B}'_2(c^*)$:
  $C^* \leftarrow (c^*, z^*, s^*)$
  If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2(C^*, \omega)$
  If $\mathcal{A}$ made an “illegal” query
    then output 0
  Else
    output $b'$
```

$\mathcal{A}$'s oracle queries are answered using the simulators H_1, H_3 and $\mathcal{O}_U$. An “illegal” oracle query is either a hash query $H_1(m, y_S^*, y_R^*, K^*)$ or $H_3(y_S^*, y_R^*, K^*)$ where $K^* = w^{*x_S^*}$, or an unsigncryption oracle query with $w = w^*$. An examination shows that

$$|\Pr[W_5^1] - \Pr[W_5^0]| \leq Adv_{\mathcal{B}'}^{\texttt{OT}}(k)\,.$$

Thus, we can conclude:

$$\begin{aligned} Adv_{\mathcal{A}}^{\texttt{in-IND}}(k) &= |\Pr[W_1^1] - \Pr[W_1^0]| \\ &\leq \frac{2q_1 + 2q_3 + 4q_U}{q} + 2Adv_{\mathcal{B}}^{\texttt{GDH}}(k) + Adv_{\mathcal{B}'}^{\texttt{OT}}(k)\,. \end{aligned}$$

□

5.2.2 Unforgeability

We are going to take a short cut to prove the unforgeability of the signcryption schemes defined by the modified Zheng transform, given the fact that we have already proven the signcryption schemes defined by the Zheng transform are insider UF-CMA secure.

Theorem 5.2.2. *Let $S = (Setup, KG_S, KG_R, SC, USC)$ be a signcryption scheme defined by the Zheng transform with parametrisation $B_1(r, e, s)$ and $B_2(r, e, s)$. Let $S' = (Setup', KG'_S, KG'_R, SC', USC')$ be the signcryption scheme defined by the modified Zheng transform with the same parametrisation. Then S' is insider UF-CMA secure if S is insider UF-CMA secure.*

Proof: Suppose there exists an adversary $\mathcal{A}$ against S' who successfully produced a forged signcryption ciphertext $C^* = (c^*, z^*, \beta_2^*)$ with the receiver's key pair (y_R^*, x_R^*). We are going to construct an adversary $\mathcal{B}$ who will use $\mathcal{A}$ as a subroutine to produce a forged signcryption ciphertext for S. Suppose we have $PP \xleftarrow{\$} Setup(1^k)$ and $(y_S^*, x_S^*) \xleftarrow{\$} KG_S(1^k)$. Then we define the adversary $\mathcal{B}$ as follows:

```
$\mathcal{B}(y_S^*)$:
  Pass $y_S^*$ to $\mathcal{A}$
  $(y_R^*, x_R^*, C'^*) \xleftarrow{\$} \mathcal{A}(y_S^*)$
  Parse $C'^*$ as $(c^*, z^*, \beta_2^*)$
  $w^* \leftarrow z^* {y_R^*}^{-\beta_2^*}$
  $r^* \leftarrow H_2(w^*)$
  $K^* \leftarrow {w^*}^{x_R^*}$
  $\tau^* \leftarrow H_3(y_S^*, y_R^*, K^*)$
  $m^* \leftarrow \mathcal{D}_{SE}(c^*, \tau^*)$
  $e^* \leftarrow H_1(m^*, y_S^*, y_R^*, K^*)$
  Compute $s^*$ from $(r^*, e^*, \beta_2^*)$
  $\beta_1^* \leftarrow B_1(r^*, e^*, s^*)$
  $C^* \leftarrow (c^*, \beta_1^*, \beta_2^*)$
  Output $(y_R^*, x_R^*, C^*)$
```

Note that $\mathcal{B}$ can access the hash functions H_1, H_2 and H_3, and the signcryption oracle $\mathcal{O}_S$. $\mathcal{A}$ should also have the access to the hash functions H'_1, H'_2 and H'_3, and the signcryption oracle $\mathcal{O}'_S$. $\mathcal{B}$ simulates H'_1, H'_2 and H'_3 by H_1, H_2 and H_3 respectively. To simulate $\mathcal{O}'_S$, $\mathcal{B}$ does the following:

```
$\mathcal{O}'_S(y_R, m)$:
  $C \leftarrow \mathcal{O}_S(y_R, m)$
  Parse $C$ as $(c, \beta_1, \beta_2)$
  $z \leftarrow g^{\beta_1}$
  $C' \leftarrow (c, z, \beta_2)$
  Output $C'$
```

Since $\mathcal{A}$'s forgery is successful, (y_R^*, x_R^*, C'^*) must satisfy the three conditions:

1. (y_R^*, x_R^*) is a valid receiver's key pair for S';
2. $m^* \neq \perp$ for $m^* \leftarrow USC'(y_S^*, x_R^*, C'^*)$; and
3. $\mathcal{A}$ never queried $\mathcal{O}'_S(y_R^*, m^*)$.

This implies

1. (y_R^*, x_R^*) is a valid receiver's key pair for S, as the keyspace are the same for S and S';
2. $USC(y_S^*, x_R^*, C^*)$ will output $m^* \neq \perp$ as H_i and H'_i are identical for $i = 1, 2, 3$;
3. $\mathcal{B}$ never queried $\mathcal{O}_S(y_R^*, m^*)$, as $\mathcal{A}$ never queried $\mathcal{O}'_S(y_R^*, m^*)$.

Hence (y_R^*, x_R^*, C^*) is a valid forgery for S. We have shown that if there exists a PPT adversary against insider UF-CMA security of S', then there exists a PPT adversary against insider UF-CMA security of S. In other words, if S is insider UF-CMA secure, then S' is insider UF-CMA secure. $\square$

5.3 Summary

In this chapter, we proposed a modified Zheng transform on meta-ElGamal signature schemes so that the resulting signcryption schemes are both IND-CCA2 secure and UF-CMA secure in the insider model, and we provided proofs of the security claims. Moreover, the cost of achieving insider security is just one extra exponentiation operation. This allows any signcryption users to choose between insider security and outsider security. Another possible benefit to having an insider secure signcryption scheme will be explored in the next chapter.

Chapter 6

One-Key Signcryption

In this chapter, we propose a transform from a two-key signcryption scheme to a one-key signcryption scheme. A two-key signcryption scheme has two independent key generation algorithms, KG_S for generating the sender's key pair, and KG_R for generating the receiver's key pair. Users of a two-key signcryption scheme need to use two independent pairs of keys for sending and receiving, respectively. A one-key signcryption scheme uses a single key generation algorithm. Users of a one-key signcryption algorithm can use the same pair of keys for both sending and receiving.

Our transform can be described as modified key concatenation. We will show that simple key concatenation results in a one-key signcryption scheme that is not IND-CCA2 secure in one-key models even if the original signcryption scheme is insider secure in two-key models. Key concatenation with our modification, however, can preserve the security features of the original signcryption scheme. We will provide formal security proofs for our claim.

6.1 From Two-Key to One-Key

We start with a clarification of the notation. We write sk_S^R to denote R's private S key, which means the receiver's private sending key. That is, we use the subscript to indicate the purpose of the key and the superscript to indicate the owner of the key.

We are now ready to describe the transform from a two-key signcryption scheme to a one-key signcryption scheme. Note that the transform is generic and we are able to prove the resulting signcryption scheme is as secure as the original one.

Definition 6.1.1. *Let $SC = (Setup, KG_S, KG_R, SC, USC)$ be a two-key signcryption scheme. We define the* one-key transform SC-1 *of SC as follows:*

```
$Setup\text{-}1(1^k)$
  $PP \xleftarrow{\$} Setup(1^k)$
  $Output\ PP$
$KG\text{-}1(PP)$
  $(sk_S, pk_S) \xleftarrow{\$} KG_S(PP)$
  $(sk_R, pk_R) \xleftarrow{\$} KG_R(PP)$
  $sk \leftarrow (sk_S, sk_R)$
  $pk \leftarrow (pk_S, pk_R)$
  $Output\ (sk, pk)$
$SC\text{-}1(sk, pk^R, m)$
  $Parse\ sk\ as\ (sk_S, sk_R)$
  $Parse\ pk^R\ as\ (pk^R_S, pk^R_R)$
  $C \xleftarrow{\$} SC(m\|pk\|pk^R, sk_S, pk^R_R) \quad (*)$
  $Output\ C$
$USC\text{-}1(sk, pk^S, C)$
  $Parse\ sk\ as\ (sk_S, sk_R)$
  $Parse\ pk^S\ as\ (pk^S_S, pk^S_R)$
  $m' \leftarrow USC(C, sk_R, pk^S_S)$
  $If\ m' = \perp$
    $halt\ and\ output\ \perp$
  $Else$
    $Parse\ m'\ as\ (m, pk', pk'')$
    $If\ (pk', pk'') = (pk^S, pk)$
      $output\ m$
    $Else$
      $output\ \perp$
```

Note that we attach both the sender's and receiver's public keys to the message m when we use SC to signcrypt in SC-1. This is crucial for preserving the security features.

Suppose we change the line with $*$ to $C \overset{\$}{\leftarrow} SC(m, pk_R^R)$, and output $m' \leftarrow USC(C, pk_S^S)$ in USC-1, then we get a one-key signcryption scheme transformed from SC by simple key concatenation. Let us call this signcryption scheme SC'.

Claim: SC' is not IND-CCA2 secure.

Proof: Suppose that the adversary $\mathcal{A}$ receives the challenge ciphertext C^* with the targeted user's public key $pk^* = (pk_S^*, pk_R^*)$ in the game $\text{Expt}_{\mathcal{A}}^{\texttt{One-in-IND-b}}(k)$ defined in Definition 2.3.13. Let $pk' = (pk_S^*, pk'_R)$ for some $pk'_R \neq pk_R^*$. The adversary then submits an unsigncryption query on (C^*, pk'). Since $pk' \neq pk^*$, $(C^*, pk') \neq (C^*, pk^*)$. Hence (C^*, pk') is a valid unsigncryption query. Due to the redundancy of pk_R^* in USC-1, the unsigncryption oracle will return m_b to the adversary when pk_R^* is replaced by some other public key pk'_R. The adversary then compares m_b with m_0 and m_1, and outputs the correct bit to win the game. Therefore, SC' is not IND-CCA2 secure. $\square$

Although it seems quite trivial that the modified key concatenation will result in a secure one-key signcryption scheme since it overcomes the problem caused by the redundancy of the targeted users's receiving public key in the unsigncryption procedure, we will provide the formal security proofs for both confidentiality and unforgeability.

Theorem 6.1.2. *Suppose SC is a two-key signcryption scheme. If SC is IND-CCA2 secure in the insider model, then SC-1 is an IND-CCA2 secure one-key signcryption scheme in the insider model.*

Proof: Let $\mathcal{A}\text{-}1 = (\mathcal{A}_1\text{-}1, \mathcal{A}_2\text{-}1)$ be a two-stage PPT adversary against the confidentiality of SC-1 in the insider model. We are going to construct a PPT adversary $\mathcal{A}$ based on $\mathcal{A}$-1 against the confidentiality of SC in the two-key insider model. We have the setup of the game as follows:

$$PP \xleftarrow{\$} Setup(1^k) \qquad (sk_R, pk_R) \xleftarrow{\$} KG_R(PP)$$

On receiving pk_R, $\mathcal{A}$ does the following:

```
Generate a key pair $(sk_S, pk_S)$
$pk \leftarrow (pk_S, pk_R)$
$(sk^*, pk^*, m'_0, m'_1, \omega) \xleftarrow{\$} \mathcal{A}_1\text{-}1(pk)$
Parse $sk^*$ as $(sk^*_S, sk^*_R)$
Parse $pk^*$ as $(pk^*_S, pk^*_R)$
$m_i \leftarrow m'_i \| pk^* \| pk$ for $i = 0, 1$
Pass $(sk^*_S, pk^*_S, m_0, m_1)$ to the challenger
Receive $C^*$ from the challenger
$b' \xleftarrow{\$} \mathcal{A}_2\text{-}1(C^*, \omega)$
Output $b'$
```

Note that the adversary $\mathcal{A}$ has access to the unsigncryption oracle $\mathcal{O}_U$. The signcryption/unsigncryption queries from $\mathcal{A}$-1 are responded to as follows:

- A signcryption query (m, pk^R):

```
Parse $pk^R$ as $(pk^R_S, pk^R_R)$
$C \xleftarrow{\$} SC(m \| pk \| pk^R, sk_S, pk^R_R)$
Output $C$
```

- An unsigncryption query (C, pk^S):

```
Parse $pk^S$ as $(pk^S_S, pk^S_R)$
$m' \leftarrow \mathcal{O}_U(C, pk^S_S)$
If $m' = \perp$
  output $\perp$
Else
  parse $m'$ as $(m, pk' \| pk'')$
  If $pk' \| pk'' = pk^S \| pk$
    output $m$
  Else
    output $\perp$
```

As described above, $\mathcal{A}$ provides $\mathcal{A}$-1 a simulated environment in which $\mathcal{A}$-1 cannot distinguish from a real one. Hence if $Adv^{\texttt{IND-CCA2}}_{\mathcal{A}\text{-}1,SC\text{-}1}$ is non-negligible, then so is $Adv^{\texttt{IND-CCA2}}_{\mathcal{A},SC}$. This proves the theorem. □

Theorem 6.1.3. *Suppose SC is a two-key signcryption scheme. If SC is UF-CMA secure in the insider model, then SC-1 is a UF-CMA secure one-key signcryption scheme in the insider model.*

Proof: Let $\mathcal{A}$-1 be a PPT adversary against the unforgeability of SC-1 in the insider model. We are going to construct a PPT adversary $\mathcal{A}$ based on $\mathcal{A}$-1 against the unforgeability of SC in the insider model. We have the setup of the game:

$$PP \xleftarrow{\$} Setup(1^k) \qquad (sk_S, pk_S) \xleftarrow{\$} KG_R(PP)$$

On receiving pk_S, $\mathcal{A}$ does the following:

```
Generate a key pair $(sk_R, pk_R)$
$pk \leftarrow (pk_S, pk_R)$
$(sk^*, pk^*, C^*) \xleftarrow{\$} \mathcal{A}\text{-}1(pk)$
Parse $sk^*$ as $(sk^*_S, sk^*_R)$
Parse $pk^*$ as $(pk^*_S, pk^*_R)$
Output $(sk^*_R, pk^*_R, C^*)$
```

Note that the adversary $\mathcal{A}$ has access to the signcryption oracle $\mathcal{O}_S$. The signcryption/unsigncryption queries from $\mathcal{A}$-1 are responded to as follows:

- A signcryption query (m', pk^R):

```
Parse $pk^R$ as $(pk^R_S, pk^R_R)$
$m \leftarrow m' \| pk \| pk^R$
Query $C \xleftarrow{\$} \mathcal{O}_S(m, pk^R_R)$
Output $C$
```

- An unsigncryption query (C, pk^S):

```
Parse $pk^S$ as $(pk_S^S, pk_R^S)$
$m' \leftarrow USC_{sk_R}(C, pk_S^S)$
If $m' = \perp$
  output $\perp$
Else
  parse $m'$ as $(m, pk' \| pk'')$
  if $pk' \| pk'' = pk^S \| pk$
    output $m$
  Else
    output $\perp$
```

As described above, $\mathcal{A}$ provides $\mathcal{A}$-1 a simulated security environment in which $\mathcal{A}$-1 cannot distinguish from a real one. Now Suppose C^* is a successful forge for SC-1 and the underlying message is m. We know that m is obtained by performing USC-1(C^*, sk^*, pk). By the construction of USC-1, we have $m' \leftarrow USC(C^*, pk_S)$, where $m' = m \| pk \| pk^*$. Since (m, pk^*) has not been queried by $\mathcal{A}$-1, (m', pk_R^*) has not been queried by $\mathcal{A}$. Hence, C^* is also a successful forge for SC. This proves the theorem.

$\square$

Combining Theorem 6.1.2 and Theorem 6.1.3, we obtain the following theorem:

Theorem 6.1.4. *If SC is a two-key signcryption scheme that is insider IND-CCA2 and insider UF-CMA secure, then SC-1 obtained from the one-key transform is a one-key signcryption scheme that is also insider IND-CCA2 and insider UF-CMA secure.*

6.2 Combined Public-Key Scheme

There is an equivalent, but more elegant, definition of the insider security of a signcryption scheme [15]. Although it is rarely used in practice, it provides a deeper understanding of the the relationship between signcryption schemes and the related concepts of public-key encryption and digital signatures.

Definition 6.2.1. *Let $SC = (Setup, KG_S, KG_R, SC, USC)$ be a signcryption scheme. We define the corresponding induced signature scheme $\mathcal{S} = (KG_{\mathcal{S}}, Sign, Verify)$ and public-key encryption scheme $\mathcal{E} = (KG_{\mathcal{E}}, \textsc{Enc}, \textsc{Dec})$ as follows:*

```
$KG_{\mathcal{S}}(1^k)$
  $PP \xleftarrow{\$} Setup(1^k)$
  $(sk_S, pk_S) \xleftarrow{\$} KG_S(PP)$
  $(sk_R, pk_R) \xleftarrow{\$} KG_R(PP)$
  $sk_{\mathcal{S}} \leftarrow (sk_S, pk_S, pk_R)$
  $pk_{\mathcal{S}} \leftarrow (sk_R, pk_S, pk_R)$
  Return $(sk_{\mathcal{S}}, pk_{\mathcal{S}})$
$Sign(m, sk_{\mathcal{S}})$
  Parse $sk_{\mathcal{S}}$ as $(sk_S, pk_S, pk_R)$
  $C \xleftarrow{\$} SC(m, sk_S, pk_R)$
  Return $C$
$Verify(m, C, pk_{\mathcal{S}})$
  Parse $pk_{\mathcal{S}}$ as $(sk_R, pk_S, pk_R)$
  $m' \leftarrow USC(C, sk_R, pk_S)$
  If $m' = m$
    return $\top$
  Else return $\perp$
```

```
$KG_{\mathcal{E}}(1^k)$
  $(sk_S, pk_S) \xleftarrow{\$} KG_S(PP)$
  $(sk_R, pk_R) \xleftarrow{\$} KG_R(PP)$
  $sk_{\mathcal{E}} \leftarrow (sk_R, pk_S, pk_R)$
  $pk_{\mathcal{E}} \leftarrow (sk_S, pk_S, pk_R)$
  Return $(sk_{\mathcal{E}}, pk_{\mathcal{E}})$
$\textsc{Enc}(m, pk_{\mathcal{E}})$
  Parse $pk_{\mathcal{E}}$ as $(sk_S, pk_S, pk_R)$
  $C \xleftarrow{\$} SC(m, sk_S, pk_R)$
  Return $C$
$\textsc{Dec}(C, sk_{\mathcal{E}})$
  Parse $sk_{\mathcal{E}}$ as $(sk_R, pk_S, pk_R)$
  $m \leftarrow USC(C, sk_R, pk_S)$
  Return $m$.
```

The signcryption scheme is insider UF-CMA secure if the induced signature scheme $\mathcal{S}$ is UF-CMA secure. The signcryption scheme is insider IND-CCA2 secure if the induced encryption scheme is IND-CCA2 secure.

Inspired by the alternative definition of insider security for signcryption, we find that it is possible to have a combined public-key scheme induced from a one-key signcryption scheme, where a combined public-key scheme is defined as follows [28][22]:

Definition 6.2.2. *A* combined public-key scheme *is a tuple of algorithms $(KG, Sign, Verify, \textsc{Enc}, \textsc{Dec})$, where $\mathcal{S} = (KG, Sign, Verify)$ is a signature scheme and $\mathcal{E} = (KG, \textsc{Enc}, \textsc{Dec})$ is a public-key encryption scheme.*

The idea of a combined public-key scheme is that a signature scheme and a public-key encryption scheme can share a key generation algorithm and hence a key pair. The security models need to capture the extra capability of an adversary caused by this convenience. We define IND-CCA2 security and UF-CMA security for a combined public-key scheme as follows:

Definition 6.2.3. *Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT two-stage adversary against a combined public-key scheme $\mathcal{C} = (KG, Sign, Verify, \textsc{Enc}, \textsc{Dec})$ with security parameter k. We define the security game $\textsc{Expt}_{\mathcal{A}}^{\textit{IND-b}}(k)$ to be:*

```
$\textsc{Expt}_{\mathcal{A}}^{\textit{IND-b}}(k)$:
  $(sk^*, pk^*) \xleftarrow{\$} KG(1^k)$
  $(m_0, m_1, \omega) \xleftarrow{\$} \mathcal{A}_1^{\mathcal{O}_\mathcal{D}, \mathcal{O}_\mathcal{S}}(pk^*)$
  $c^* \xleftarrow{\$} \textsc{Enc}(m_b, pk^*)$
    If $|m_0| \neq |m_1|$ then $C^* \leftarrow \perp$
  $b' \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}_\mathcal{D}, \mathcal{O}_\mathcal{S}}(c^*, \omega)$
  Output $b'$ ,
```

where the decryption oracle $\mathcal{O}_\mathcal{D}$ and the signing oracle $\mathcal{O}_\mathcal{S}$ are defined as

$$\mathcal{O}_\mathcal{D}(c) = \textsc{Dec}(c, sk^*) \quad \textit{and} \quad \mathcal{O}_\mathcal{S}(m) = Sign(m, sk^*) \ ,$$

with the condition that $\mathcal{A}_2$ cannot query $\mathcal{O}_\mathcal{D}(c^*)$. The adversary's advantage is defined to be

$$Adv_{\mathcal{A}}^{\textit{IND}}(k) = |\Pr[\textsc{Expt}_{\mathcal{A}}^{\textit{IND-1}}(k) = 1] - \Pr[\textsc{Expt}_{\mathcal{A}}^{\textit{IND-0}}(k) = 1]|.$$

The combined public-key encryption scheme is said to be IND-CCA2 secure *if $Adv_{\mathcal{A}}^{\textit{IND}}(k)$ is negligible in k for every PPT adversary $\mathcal{A}$.*

Definition 6.2.4. *Let $\mathcal{A}$ be a PPT adversary against a combined public-key scheme $\mathcal{C} = (KG, Sign, Verify, \textsc{Enc}, \textsc{Dec})$ with security parameter k. We define the security game $\textsc{Expt}_{\mathcal{A}}^{\textit{UF}}(k)$ to be:*

```
$\text{Expt}^{\mathit{UF}}_{\mathcal{A}}(k)$:
  $(sk^*, pk^*) \xleftarrow{\$} KG(1^k)$
  $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathcal{S}}, \mathcal{O}_{\mathcal{D}}}(pk^*)$
  Output 1 if
    (a) $\top \leftarrow Verify(m^*, \sigma^*, pk^*)$
    (b) $\mathcal{A}$ never queried $\mathcal{O}_{\mathcal{S}}(m^*)$
  Else output 0
```

where the decryption oracle $\mathcal{O}_{\mathcal{D}}$ and the signing oracle $\mathcal{O}_{\mathcal{S}}$ are defined as

$$\mathcal{O}_{\mathcal{D}}(c) = \text{Dec}(c, sk^*) \quad \textit{and} \quad \mathcal{O}_{\mathcal{S}}(m) = Sign(m, sk^*) \ .$$

The adversary's advantage is defined to be

$$Adv^{\mathit{UF}}_{\mathcal{A}}(k) = \Pr[\text{Expt}^{\mathit{UF}}_{\mathcal{A}}(k) = 1].$$

The combined public-key scheme is said to be UF-CMA *secure if $Adv^{\mathit{UF}}_{\mathcal{A}}(k)$ is negligible for every PPT adversary $\mathcal{A}$.*

Inspired by the alternative definition of insider security for signcryption, we find that it is possible to have a secure combined public-key scheme induced from an insider secure one-key signcryption scheme. To formalise this idea, we first define the combined public-key scheme induced from a one-key signcryption scheme.

Definition 6.2.5. *Let $SC = (Setup, KG, SC, USC)$ be a one-key signcryption scheme. We define the corresponding combined public-key scheme $\mathcal{C} = (KG_{\mathcal{C}}, Sign, Verify, \text{Enc}, \text{Dec})$ as follows:*

```
$KG_{\mathcal{C}}(1^k)$
  $PP \xleftarrow{\$} Setup(1^k)$
  $(sk^A, pk^A) \xleftarrow{\$} KG(PP)$
  $(sk^U, pk^U) \xleftarrow{\$} KG(PP)$
  $sk \leftarrow (sk^A, pk^A, pk^U)$
  $pk \leftarrow (sk^U, pk^A, pk^U)$
  Return $(sk, pk)$
```

```
$Sign(m, sk)$
  $Parse\ sk\ as\ (sk^A, pk^A, pk^U)$
  $C \xleftarrow{\$} SC(m, sk^A, pk^U)$
  $Return\ C$
$Verify(m, C, pk)$
  $Parse\ pk\ as\ (sk^U, pk^A, pk^U)$
  $m' \leftarrow USC(C, sk^U, pk^A)$
  $If\ m' = m$
    $return\ \top$
  $Else\ return\ \bot$
```

```
$\textsc{Enc}(m, pk)$
  $Parse\ pk\ as\ (sk^U, pk^A, pk^U)$
  $C \xleftarrow{\$} SC(m, sk^U, pk^A)$
  $Return\ C$
$\textsc{Dec}(C, sk)$
  $Parse\ sk\ as\ (sk^A, pk^A, pk^U)$
  $m \leftarrow USC(C, sk_A, pk^U)$
  $Return\ m.$
```

It is worth noting that KG is run twice in $KG_{\mathcal{C}}$. The first execution of KG is to generate a pair of keys for the user A of the signcryption scheme. The second execution of KG is to generate a pair of keys for some other user U of the signcryption scheme. There are two interpretation of the user U.

The signcryption schemes in our thesis are not broadcasting signcryption schemes. That is, any communication that uses a signcryption scheme is between one user and another user, instead of a group of users. However, signature schemes are designed so that whoever knows the public key of the signer is able to verify signatures from that signer, and public-key encryption schemes are designed so that whoever knows the public key of the receiver is able to send encrypted messages to that receiver.

One way to overcome this is to acquire each verifier's public key when signing a message, and whoever wants to sending encrypted message to A has to have a pair of keys known to A. By doing this, it is necessary to specify the other user U whenever A is signing or receiving encrypted messages.

Another way to solve the problem is to have a universal user U of the signcryption scheme. This is like a wild card. The key pair of U is publicly known to everyone. This is when insider security becomes useful, since it assumes that the adversary is a legitimate user of a signcryption scheme. Intuitively, if a signcryption scheme is secure against an insider, then it should be secure if there is a universal user like U, and hence our combined public-key scheme is secure. A formal security proof is presented as follows:

Theorem 6.2.6. *Let $SC = (Setup, KG, SC, USC)$ be a one-key signcryption scheme that is insider IND-CCA2 and UF-CMA secure. Let $\mathcal{C} = (KG_{\mathcal{C}}, Sign, Verify, \textsc{Enc}, \textsc{Dec})$ be the corresponding combined public-key scheme induced from SC. Then $\mathcal{C}$ is IND-CCA2 and UF-CMA secure.*

Proof: We first show that $\mathcal{C}$ is IND-CCA2 secure.

Suppose that there is a PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that breaks IND-CCA2 security for the combined public-key scheme $\mathcal{C}$. Then we claim that there is PPT adversary $\mathcal{A}$ breaking one-key insider IND-CCA2 security for the signcryption scheme SC. To construct $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we consider the following setting:

$$PP \xleftarrow{\$} Setup(1^k) \quad \text{and} \quad (sk^T, pk^T) \xleftarrow{\$} KG(PP) \ ,$$

where (sk^T, pk^T) is the key pair of the targeted user.

The construction is described as follows:

- On receiving pk^T, the adversary $\mathcal{A}_1$ generates a key pair (sk^U, pk^U), and passes $pk^* \leftarrow (sk^U, pk^T, pk^U)$ to $\mathcal{B}_1$.
- On receiving pk^*, the adversary $\mathcal{B}_1$ can query a signing oracle $\mathcal{O}_{\mathcal{S}}$ and a decryption query $\mathcal{O}_{\mathcal{D}}$, where both oracles are simulated by $\mathcal{A}_1$ using a signcryption oracle $\mathcal{O}_S$ and an unsigncryption oracle $\mathcal{O}_U$, as follows:

```
$\mathcal{O}_{\mathcal{S}}(m)$                          $\mathcal{O}_{\mathcal{D}}(C)$
  $C \xleftarrow{\$} \mathcal{O}_S(m, pk^U)$                 $m \leftarrow \mathcal{O}_U(C, pk^T)$
  Return $C$                          Return $m$.
```

- Having made enough queries, $\mathcal{B}_1$ outputs (m_0, m_1, ω). Accordingly, $\mathcal{A}_1$ outputs $(m_0, m_1, sk^U, pk^U, \omega)$.

- After receiving C^* from the challenger and some information ω from $\mathcal{A}_1$, $\mathcal{A}_2$ passes (C^*, ω) to $\mathcal{B}_2$.
- $\mathcal{B}_2$ can query the signing oracle and the decryption oracle, which are both simulated by the signcryption oracle and the unsigncryption oracle as above.
- At the end, $\mathcal{B}_2$ outputs a bit b'. $\mathcal{A}_2$ sees b' and also outputs b'.

We now analyse the probability that $\mathcal{A}$ wins the game. Suppose $\mathcal{B}$ is successful. Then, we have:

- $m_{b'} = \textsc{Dec}(C^*)$, and
- $\mathcal{B}_2$ has never queried $\mathcal{O}_{\mathcal{D}}(C^*)$.

This implies:

- $m_{b'} = USC(C^*, sk^T, pk^U)$, and
- $\mathcal{A}_2$ has never queried $\mathcal{O}_U(C^*, pk^U)$.

Therefore,

$$Adv_{\mathcal{A}}^{\texttt{One-in-IND}} \geq Adv_{\mathcal{B}}^{\texttt{IND}}(k) \; .$$

In other words, if $\mathcal{B}$ is a PPT adversary with non-negligible advantage against IND-CCA2 security of the combined public-key scheme $\mathcal{C}$, then $\mathcal{A}$ is a PPT adversary with non-negligible advantage against one-key insider IND-CCA2 security of the signcryption scheme SC. By our assumption that SC is a one-key insider IND-CCA2 secure signcryption scheme, we can conclude that $\mathcal{C}$ is a IND-CCA2 secure combined public-key scheme.

The next task is to prove that $\mathcal{C}$ is UF-CMA secure.

Suppose that there is a PPT adversary $\mathcal{B}$ that breaks UF-CMA security for the combined public-key scheme $\mathcal{C}$. Then we claim that there is PPT adversary $\mathcal{A}$ breaking one-key insider UF-CMA security for the signcryption scheme SC. To construct $\mathcal{A}$, we consider the following setting:

$$PP \overset{\$}{\leftarrow} \mathit{Setup}(1^k) \quad \text{and} \quad (sk^T, pk^T) \overset{\$}{\leftarrow} KG(PP) \ ,$$

where (sk^T, pk^T) is the key pair of the targeted user.

The construction is described as following:

- On receiving pk^T, the adversary $\mathcal{A}$ generates a key pair (sk^U, pk^U), and passes $pk^* \leftarrow (sk^U, pk^T, pk^U)$ to $\mathcal{B}$.
- On receiving pk^*, the adversary $\mathcal{B}$ can query a signing oracle $\mathcal{O}_{\mathcal{S}}$ and a decryption query $\mathcal{O}_{\mathcal{D}}$, where both oracles are simulated by $\mathcal{A}$ using a signcryption oracle $\mathcal{O}_S$ and an unsigncryption oracle $\mathcal{O}_U$ as following:

```
$\mathcal{O}_{\mathcal{S}}(m)$
  $C \overset{\$}{\leftarrow} \mathcal{O}_S(m, pk^U)$
  Return $C$
```

```
$\mathcal{O}_{\mathcal{D}}(C)$
  $m \leftarrow \mathcal{O}_U(C, pk^T)$
  Return $m$.
```

- Having made enough queries, $\mathcal{B}$ outputs (m^*, C^*). Accordingly, $\mathcal{A}$ outputs (m^*, C^*).

Suppose $\mathcal{B}$ successfully produces a forgery (m^*, C^*). Then, we have:

- $\top \leftarrow \mathit{Verify}(m^*, C^*, pk^*)$, and
- $\mathcal{A}$ never queried $\mathcal{O}_{\mathcal{S}}(m^*)$.

This implies:

- $m^* \neq \perp$ for $m^* \leftarrow USC(pk^U, sk^T, C^*)$, and
- $\mathcal{A}$ never queried $\mathcal{O}_S(pk^U, m^*)$.

Therefore,

$$Adv_{\mathcal{A}}^{\texttt{One-in-UF}} \geq Adv_{\mathcal{B}}^{\texttt{UF}}(k) \ .$$

In other words, if $\mathcal{B}$ is a PPT adversary with non-negligible advantage against UF-CMA security of the combined public-key scheme $\mathcal{C}$, then $\mathcal{A}$ is a PPT adversary with non-negligible advantage against one-key insider IND-CCA2 security of the signcryption scheme SC. By our assumption that SC is a one-key insider UF-CMA secure signcryption scheme, we can conclude that $\mathcal{C}$ is a UF-CMA secure combined public-key scheme.

$\square$

6.3 Summary

We showed that a one-key signcryption scheme with insider security can induce a secure combined public-key scheme. In Chapter 5, we demonstrated that the modified Zheng transform can produce insider secure signcryption schemes from a large class of meta-ElGamal signature schemes. In this chapter, we showed that any insider secure signcryption scheme can be turned into a one-key insider secure signcryption scheme at a cost of key repetition. Hence, we have a large class of signcryption schemes that are one-key insider secure. Moreover, all of them can induce a secure combined public-key scheme, a pair of a signature scheme and a public-key encryption scheme that can securely share the same key pair.

Chapter 7

Conclusion and Future Work

This thesis has focused on the study of provable security of signcryption schemes that are obtained from generalisation of existing signcryption schemes.

We proposed two generic transforms from a meta-ElGamal signature scheme to a signcryption scheme: the Zheng transform and the Gamage transform, and provided security proofs to show that the resulting signcryption schemes are outsider confidential and insider unforgeable. We then modified the Zheng transform so that the resulting signcryption schemes can achieve insider confidentiality as well as insider unforgeability. We also proposed a transform from a two-key signcryption scheme to a one-key signcryption scheme. While all previous transforms are meta-ElGamal based, the last transform is generic, and can be applied to any two-key signcryption schemes. Moreover, we showed that an insider secure one-key signcryption scheme can induce a secure combined public-key scheme, where a signature scheme and a public-key encryption scheme can securely share the same key pair.

7.1 Future Work

Signcryption, as a cryptographic primitive, is still relatively immature. There are still many topics in signcryption waiting to be explored. We list some possible future work as follows:

- For both meta-ElGamal signature schemes and meta-ElGamal signcryption schemes, the diversity comes from the variation of the tuple of functions (B_1, B_2, B_3) that are used in the equation:

  $$B_1 = xB_2 + tB_3 \ .$$

  One possible research direction is to consider modifications to the equation, so that designs of alternative signature schemes or signcryption schemes could be explored.

- We required four properties on B_1 and B_2 as in Section 4.3. The security proofs depend on the third and fourth properties. It is worth noting that we cannot provide security proofs or attacks for the case that B_1 or B_2 do not satisfy the properties. It may be interesting to find an attack or security proofs for the case that B_1 or B_2 do not satisfy the third or fourth properties.

- It would be interesting to try to find a modification of the Gamage transform so that the resulting signcryption schemes can achieve insider confidentiality and unforgeability.

- We have shown that an insider secure one-key signcryption scheme induces a secure combined public-key scheme as a generic result. However, we do not know whether the reverse is true, i.e., whether a secure combined public-key scheme can induce an insider secure one-key signcryption scheme.

Bibliography

[1] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumption and an analysis of DHIES. In *The Cryptographer's Track at RSA Conference 2001– CT-RSA '01*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer-Verlag, 2001.

[2] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *International Conference on the Theory and Applications of Cryptographic Techniques – EuroCrypt '02*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[3] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In *5th International Workshop on Practice and Theory in Public Key Cryptosystems – PKC '02*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer-Verlag, 2002.

[4] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, 2007.

[5] M. Barbosa and P. Farshim. Certificateless signcryption. In *ACM Symposium on Information, Computer and Communications Security – AsiaCCS '08*, pages 369–372. ACM, 2008.

[6] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *13th ACM Conference on Computer and Communications Security – CCS '06*, pages 390–399. ACM, 2006.

[7] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security – CCS '93*, pages 62–73. ACM, 1993.

[8] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *24th International Conference on the Theory and Applications of Cryptographic Techniques – EuroCrypt '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer-Verlag, 2006.

[9] T. E. Bjørstad and A. W. Dent. Building better signcryption schemes with tag-KEMs. In M. Yung, Y. Dodis, A. Kiayas, and T. Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 491–507. Springer-Verlag, 2006.

[10] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51:557–594, 2004.

[11] L. Chen and J. Malone-Lee. Improved identity-based signcryption. In *8th International Workshop on Theory and Practice in Public Key Cryptography – PKC '05*, volume 3386 of *Lecture Notes in Computer Science*, pages 362–379. Springer-Verlag, 2005.

[12] A. W. Dent. Fundamental problems in provable security and cryptography. *Philosophical Transactions of the Royal Society*, 364(1849):3215–3230, 2006.

[13] A. W. Dent. A brief history of provably-secure public-key encryption. In *1st International Conference on Progress in Cryptology – AfricaCrypt '08*, volume 5023 of *Lecture Notes in Computer Science*, pages 357–370. Springer-Verlag, 2008.

[14] A. W. Dent, M. Fischlin, M. Manulis, M. Stam, and D. Schröder. Confidential signatures and deterministic signcryption. In *13th International Conference on Practice and Theory in Public Key Cryptography – PKC '10*, volume 6056 of *Lecture Notes in Computer Science*, pages 462–479. Springer-Verlag, 2010.

[15] A. W. Dent and Y. Zheng, editors. *Practical Signcryption.* Springer-Verlag, 2010.

[16] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[17] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in cryptology – Crypto '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[18] C. Gamage, J Leiwo, and Y. Zheng. Encrypted message authentication by firewalls. In *2nd International Workshop on Practice and Theory in Public Key Cryptography – PKC '99*, volume 1560 of *Lecture Notes in Computer Science*, pages 69–81. Springer-Verlag, 1999.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28:270–299, 1984.

[20] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[21] J. A. Gregg. *On Factoring Integers and Evaluating Discrete Logarithms.* Harvard University, 2003.

[22] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In *8th ACM conference on Computer and Communications Security – CCS '01*, pages 215–224. ACM, 2001.

[23] P. Horster, H. Petersen, and M. Michels. Meta-ElGamal signature schemes. In *2nd ACM Conference on Computer and Communications Security – CCS '94*, pages 96–107. ACM, 1994.

[24] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman and Hall CRC, 2007.

[25] B. Libert and J.-J. Quisquater. A new identity based signcryption scheme from pairings. In *IEEE Information Theory Workshop*, pages 155–158. IEEE Information Theory Society, 2003.

[26] B. Libert and J.-J. Quisquater. Improved signcryption from q-Diffie-Hellman problems. In *4th International Conference on Security in Communication Networks – SCN '04*, volume 3352 of *Lecture Notes in Computer Science*, pages 220–234. Springer-Verlag, 2005.

[27] J. Malone-Lee. Signcryption with non-interactive non-repudiation. *Designs, Codes and Cryptography*, 37(1):81–109, 2005.

[28] K. G. Paterson, J. C. N. Schuldt, M. Stam, and S. Thomson. On the joint security of encryption and signature, revisited. In *17th International Conference on the Theory and Application of Cryptology and Information Security – AsiaCrypt '11*, volume 7073 of *Lecture Notes in Computer Science*, pages 161–178. Springer-Verlag, 2011.

[29] J. Pieprzyk and D.Pointcheval. Parallel authentication and public-key encryption. In *8th Australasian Conference on Information Security and Privacy – ACISP'03*, volume 2727 of *Lecture Notes in Computer Science*, pages 387–401. Springer-Verlag, 2003.

[30] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[31] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1979.

[32] C. P. Schnorr. Efficient signature generation for smart cards. In *Advances in Cryptology – Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 1989.

[33] J.-B. Shin, K. Lee, and K. Shim. New DSA-verifiable signcryption schemes. In *5th International Conference on Information Security and Cryptology – ICISC'02*, volume 2587 of *Lecture Notes in Computer Science*, pages 35–47. Springer-Verlag, 2003.

[34] V. Shoup. Sequences of games: A tool for taming complexity in security proofs. Available from `http://eprint.iacr.org/2004/332/`, 2004.

[35] R. Steinfeld and Y. Zheng. A signcryption scheme based on integer factorization. In *3rd International Workshop on Information Security – ISW '00*, volume 1975 of *Lecture Notes in Computer Science*, pages 308–322. Springer-Verlag, 2000.

[36] J. Talbot and D.J.A. Welsh. *Complexity and Cryptography: An Introduction.* Cambridge University Press, 2006.

[37] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) $\ll$ cost(signature) + cost(encryption). In *Advances in Cryptology – Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.

[38] Y. Zheng. Identification, signature and signcryption using high order residues modulo an RSA composite. In *4th International Workshop on Practice and Theory in Public Key Cryptography – PKC '01*, volume 1992 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 2001.