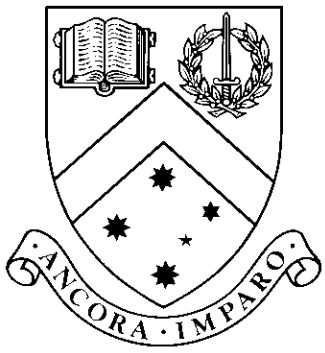
Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting

by

Joonsang Baek



Dissertation

Submitted by Joonsang Baek

for fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

in the School of School of Network Computing at

Monash University

Monash University

January, 2004

To my parents

Contents

List of Figures

Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting

Joonsang Baek, PhD
Monash University, 2004

Abstract

This thesis presents two main themes. One is the study of existing cryptographic schemes through a new perspective and the other is the construction of new schemes which will enhance the functionality of current public key cryptography. Specifically, contributions of this thesis are as follows:

We show that the slightly modified version of one of Zheng and Seberry's public key encryption schemes presented at Crypto '92 is secure against chosen ciphertext attack in the random oracle model, relative to, in fact, the Gap Diffie-Hellman problem.

We present strong and realistic confidentiality and unforgeability notions for generic signcryption schemes, which encompass the conventional notions of indistinguishability against chosen ciphertext attack and existential unforgeability against chosen message attack. We then show that Zheng's original signcryption scheme meets our confidentiality and unforgeability notions in the random oracle model, relative to the Gap Diffie-Hellman problem for confidentiality and the standard Discrete-Logarithm problem for unforgeablity.

We construct the first identity-based threshold decryption scheme secure against chosen ciphertext attack. A formal proof of security of the scheme is provided in the random oracle model, assuming the Bilinear Diffie-Hellman problem is computationally hard. We also construct, by extending the proposed identity-based threshold decryption scheme, a mediated identity-based encryption scheme secure against more powerful attacks than those considered previously.

We formalize the concept of identity-based threshold signature and give the first provably secure scheme based on the bilinear pairings. Like our identity-based threshold decryption scheme, an important feature of the proposed identity-based threshold signature scheme is that the private key associated with an identity is shared among signature generation servers, which, we claim, is more practical than sharing a master key of the Public Key Generator.

Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Joonsang Baek
January 6, 2004

Acknowledgments

I would like to express my sincere gratitude to my supervisor Dr Yuliang Zheng. I very much appreciate his constant support and encouragement. With his thoughtful guidance, I was able to stay focused during the entire period of my PhD. I am also deeply grateful to my co-supervisor Dr Jan Newmarch for his encouragement and support. Weekly meetings with him have been delightful experiences for understanding the real-world security problems and enhancing my knowledge of other areas of computing. I would like to extend my gratitude to Dr Lee Seldon, who helped me to have great teaching experiences in my final year at Monash.

Special thanks go to my friend and former colleague Ron Steinfeld. I have benefited greatly from his encouragement, inspiration, and friendship. Collaborating him was a real pleasure.

Warm gratitude also goes to my friends at Monash including Craig MacDonald, Wei Ye, Samantha Senaratna, Hugo LeRoux, Minh Le Viet, Kei Nam Tsoi, Adrian Ryan, Paulo Tam, Nisha Roy, Robin Kirk, Chu Tiong Yeoh, Benny Nasution, Robert Bram, Yandong Fan, How Keat Low, and Yongseok Choi.

Many thanks to Michael Scriven and Shirley Scriven for their generosity while my wife and I stay in their rear house.

I cannot thank my father and mother enough for their great love and spiritual support. I also thank my brother and sister very much for their encouragement.

Last, but not least, I would like to express my deepest thanks to my wife Hyung-Ran. Without her support, tolerance, and love, this thesis would not have been possible.

Joonsang Baek

Monash University
January 2004

Publications

Publications arising from this thesis include:

J. Baek, R. Steinfeld, and Y. Zheng (2002), *Formal Proofs for the Security of Signcryption*, Public Key Cryptography – Proceedings of PKC 2002, Lecture Notes in Computer Science 2274, pages 80-98, Springer-Verlag, 2002.

R. Steinfeld, J. Baek, and Y. Zheng (2002), *On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes*, Australasian Conference on Information Security and Privacy – Proceedings of ACISP 2002, Lecture Notes in Computer Science 2384, pages 241–256, Springer-Verlag, 2002.

J. Baek and Y. Zheng (2003), *Zheng and Seberry's Public Key Encryption Scheme Revisited*, International Journal of Information Security (IJIS), Vol. 2, No. 1, pages 37–44, Springer-Verlag, 2003.

J. Baek, R. Steinfeld, and Y. Zheng (2003), *Formal Proofs for the Security of Signcryption (Full Version)*, Submitted to Journal of Cryptology.

J. Baek and Y. Zheng (2003), *Simple and Efficient Threshold Cryptosystem from the Gap Diffie-Hellman Group*, IEEE Global Communications Conference – Proceedings of IEEE GLOBECOM Conference, Communication Security Track, SC04-7, pages 1491–1495, IEEE, 2003.

J. Baek and Y. Zheng (2004), *Identity-Based Threshold Decryption*, Public Key Cryptography – Proceedings of PKC 2004, Lecture Notes in Computer Science, Springer-Verlag, 2004, to appear.

J. Baek and Y. Zheng (2004), *Identity-Based Threshold Signature from the Bilinear Pairings*, IEEE International Conference on Information Technology: Coding and Computing – Proceedings of ITCC 2004, Information Assurance and Security Track, IEEE Computer Society, 2004, to appear.

Permanent Address: School of Network Computing
Monash University
Australia

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Glenn Maughan and modified by Dean Thompson and David Squire of Monash University.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Cryptographic Schemes and Their Security

Over the last decades, the deployment of the Internet and distributed systems has significantly simplified the exchange of information between remote users. However, this in turn, has led to an explosive growth in *threats* over the networks such as electronic eavesdropping, fraud, and malicious code. In order to protect valuable resources from such threats, engineers have been developing diverse security systems.

As a base technology, *cryptography* is playing a central role in building security systems. Hence, for cryptographers, it is important to design robust and versatile cryptographic schemes which can serve as sound security primitives. However, this task is not always easy in that a cryptographic scheme which looks sturdy at first sight sometimes turns out to have serious security flaws. A good example is Bleichenbacher's [22] attack on the RSA encryption standard PKCS #1 which was implemented in the widely-used Secure Socket Layer (SSL) protocol. His attack indeed surprised the community as the cryptographic schemes had generally been believed to be the strongest link in the security system. It should be emphasized, however, that what Bleichenbacher attacked is not the RSA [103] function which is related to the problem of factoring a large integer (the underlying computational primitive of the PKCS #1), but, the *way of using it* to construct a public key encryption scheme. It seems that the attacked PKCS #1 had been constructed through intuitive heuristics rather than a rigorous *analysis*. Consequently, Bleichenbacher's attack suggests that the heuristic approach

to design cryptographic schemes can be risky and the security of cryptographic products should be evaluated very carefully before they are deployed.

However, a sound approach to evaluating the security of cryptographic schemes or protocols already exists. This approach is called "provable security" and stems from Goldwasser and Micali's [62] pioneering work on public key encryption schemes that hides all partial information about plaintext. According to Stinson [119], the provable security approach can be described as follows:

> "*This approach is to provide evidence of security by reducing the security of the cryptosystem to some well-studied problem that is thought to be difficult. For example, it may be able to prove a statement of the type 'a given cryptosystem is secure if a given integer n cannot be factored.' Cryptosystems of this type are sometimes called provably secure.*"

That is, in the provable security approach, one ensures the security of a given cryptographic scheme by presenting a "reduction" between the properly defined security notion for the scheme and the underlying primitive such as RSA or DES [84], in a similar way as proving that a given computational problem is NP-complete in the theory of computation.

Since Goldwasser and Micali's work, there have been a number of research works taking this approach and it has become a paradigm of cryptographic research. As a consequence, and possibly affected by the negative results on the security of the past cryptographic standards, e.g, [23], [64], and [22], today's standard organizations such as ISO-IEC [109], P1363 [67], and NESSIE [98] strongly recommend that a precise security analysis based on the provable security approach should be included in a proposal of new cryptographic schemes or protocols.

1.1.2 Scope of This Thesis

The provable security approach, as discussed above, is not only important by itself to analyze the security of a given cryptographic scheme rigorously but also helps to design new ones, with a high level of security guarantee. With this in mind, we pursue two goals in this thesis:

- One goal is to *analyze* the security of important existing cryptographic schemes whose security has not been analyzed in a framework of the provable security approach. To this end, we have selected Zheng and Seberry's public key encryption [130] and Zheng's original signcryption [124] schemes

which have been extensively explored by a number of research works since they were proposed. Taking the provable security approach, we rigorously analyze the security of these schemes in Chapters 3 and 4.

- The other goal is to *construct* new cryptographic schemes which will enhance the functionality of identity-based cryptography, specific public key cryptography that does not depend on the public key directories. The concept of identity-based cryptography was proposed in 1984 by Shamir [108], but has flourished only recently thanks to the recent work of Boneh and Franklin [27]. In Chapters 5 and 6, we treat the problem of giving identity-based cryptography the functionality of threshold cryptography whose main motivation is to decentralize the power of cryptographic operations [44].

In the next section, we overview the subject matter of each chapter in relation to achieving the above two goals.

1.2 Overview of Chapters

1.2.1 Chapter 2: Background

In this chapter, we survey the background theory on which the subject matter of the rest of the thesis is based. First, we review the basics of public key encryption, digital signature, and identity-based encryption/signature. We also review some computational primitives such as the Integer Factorization, Discrete-Logarithm, and various Diffie-Hellman problems. We then study the important security notions for public key cryptographic schemes such as the indistinguishability of encryptions [62] against chosen ciphertext attack [85, 100, 48, 16] and unforgeability against chosen message attack [63]. Finally, we discuss the random oracle model [20], which is somewhat controversial but is an important ingredient of the *practice-oriented provable security* paradigm in which one can design *efficient* provably-secure cryptographic schemes [14].

1.2.2 Chapter 3: Security Analysis of Zheng and Seberry's Public Key Encryption Scheme

This chapter is devoted to the security analysis of Zheng and Seberry's public key encryption scheme presented at Crypto '92 [130], which has affected the design of many public key encryption schemes proposed since then. We show

that the slightly modified version of this scheme proposed by Lim and Lee [77] is indeed *provably secure* against chosen ciphertext attack in the random oracle model, relative to the "Gap Diffie-Hellman" problem [90]. (Note that the Gap Diffie-Hellman problem is to solve the Computational Diffie-Hellman problem [45] with the help of the oracle (algorithm) for solving the Decisional Diffie-Hellman problem.) Interestingly, this result contradicts the recent security proof claimed by Soldera, Seberry, and Qu (SSQ) [115] that only the intractability of the Decisional Diffie-Hellman problem is sufficient for the modified version of Zheng and Seberry's scheme to be secure against chosen ciphertext in the random oracle model. Through informal and formal arguments, we show that their claim is in fact false.

Publication Information. An earlier version of the results in this chapter was published in International Journal of Information Security (IJIS) [8]. A part of discussions on SSQ's result was motivated by the paper presented at Australasian Conference on Information Security and Privacy 2002 (ACISP 2002) [117].

1.2.3 Chapter 4: Security Analysis of Signcryption

Signcryption is a public key cryptographic scheme that provides simultaneously both message confidentiality and unforgeability at a low computational and communication overhead, which was originally proposed by Zheng at Crypto '97 [124]. Like Zheng and Seberry's public key encryption scheme, Zheng's original signcryption scheme has not been analyzed within the framework of provable security. This chapter deals with the problem of formulating security models for signcryption and analyzing the security of Zheng's original scheme. After presenting realistic confidentiality and unforgeability models for (general) signcryption which encompass chosen ciphertext and chosen message attacks, we show that Zheng's original signcryption scheme is *provably secure in our confidentiality model* relative to the Gap Diffie-Hellman problem and is *provably secure in our unforgeability model* relative to the standard Discrete-Logarithm problem. All these results are shown in the random oracle model.

Publication Information. An earlier version of the results in this chapter was presented at International Workshop on Practice and Theory in Public Key Cryptography 2002 (PKC 2002) [6].

1.2.4 Chapter 5: Identity-Based Threshold Decryption from the Bilinear Map

Threshold decryption [44] is of particular importance where the decentralization of the power to decrypt is required. The motivation for identity-based encryption originally proposed by Shamir [108] is to provide confidentiality without the need of exchanging public keys or keeping public key directories. A major advantage of ID-based encryption is that it allows one to encrypt a message by using a recipient's identifiers such as an email address. This chapter focuses on the problem of constructing an *identity-based threshold decryption scheme* in which threshold decryption is realized in the identity-based setting. First, we carefully examine issues related to the construction of such a scheme and argue that it is important in practice to design an identity-based threshold decryption scheme in which a private key associated with an identity is shared rather than a master key of the PKG. We then present the first identity-based threshold decryption scheme secure against chosen ciphertext attack. A formal proof of security of this scheme is provided in the random oracle model, assuming the Bilinear Diffie-Hellman problem [27] is computationally hard. Another contribution of this chapter is, by extending the proposed identity-based threshold decryption scheme, to construct a mediated identity-based encryption scheme [46] secure against more powerful attacks than those considered previously.

Publication Information. An earlier version of the results in this chapter has been accepted to present at International Workshop on Practice and Theory in Public Key Cryptography 2004 (PKC 2004) [10]. A part of the results in this chapter was motivated by our paper presented at Communication Security Track of IEEE 2003 Global Communication Conference (GLOBECOM 2003) [9].

1.2.5 Chapter 6: Identity-Based Threshold Signature from the Bilinear Map

In this chapter, we formalize the concept of identity-based threshold signature and give the first provably secure scheme based on the bilinear pairings. Like our identity-based threshold decryption scheme presented in Chapter 5, an important feature of this scheme is that a private key associated with an identity is shared among a number of signature generation servers rather than a master key of the PKG. From a theoretical point of view, an interesting aspect of our results is that the security of one of the proposed verifiable secret-sharing schemes used to construct our identity-based threshold signature scheme is relative to the modified

Generalized Bilinear Inversion (mGBI) problem, which is a slight modification of the Generalized Tate Inversion (GTI) problem that Joux [69] recently proposed and questioned how it can be applied to build cryptographic protocols. Hence, our result gives a partial answer to Joux's question.

Publication Information. An earlier version of the results in this chapter has been accepted to present at Information Assurance and Security Track of IEEE 2004 International Conference on Information Technology: Coding and Computing (ITCC 2004) [11].

Chapter 2

Background

2.1 Introduction

Public key cryptography has made security services over the networks more versatile than ever before. Using public key cryptography, the end-users who have never met each other before can now exchange data securely and conveniently. Moreover, public key cryptography has made it possible to realize secure electronic commerce such as secure fund transfers and digital cash. Indeed, a large number of today's security applications over the Internet are built on it.

In this chapter, we review some basic concepts from public key cryptography such as:

- motivations for public key encryption, digital signature, and identity-based encryption/signature schemes and their descriptions;
- well-known computational problems on which the current public key cryptographic schemes are based;
- fundamental security notions and the random oracle model [20].

Since the aim of this chapter is to give an overview of the background theory that will be used in later chapters, the above subject matter will be treated in a rather informal way. The definitions given in this chapter will be revisited in a more formal way from Chapter 3.

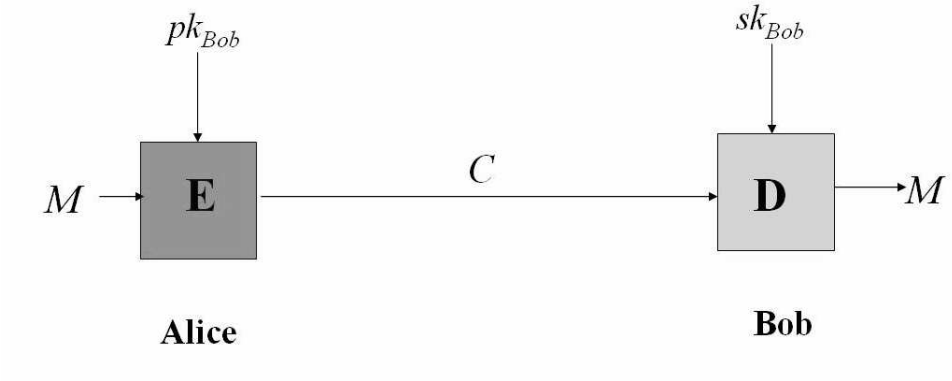


Figure 2.1: Public Key Encryption

2.2 Basic Public Key Cryptographic Schemes

The concept of public key cryptography was first introduced by Diffie and Hellman [45] in 1976. The main motivation for public key cryptography is to remove the burden of key sharing in the conventional symmetric key cryptography in which a separate secret key is needed for each pair of users to communicate in private. More precisely, if there are n users who want to exchange secret data using the symmetric key cryptography, $n(n-1)/2$ keys are needed and this number increases rapidly as the number of users grows. Yet, in public key cryptography, each user creates a pair of keys, one of which is to be publicized while the one is to be kept secret. The publicized key, referred to as "public key", is used as encryption key, but the secret key, referred to as "private key", is used as decryption key. As a result, there is no key sharing problem as in symmetric key cryptography. Another remarkable achievement of public key cryptography is that one can construct a digital signature scheme by using the private key as signature generation key while using the public key as verification key.

2.2.1 Public Key Encryption

A public key encryption scheme is one of the fundamental public key cryptographic schemes and can be described as follows.

- Key Generation: The receiver Bob creates his private and public key pair, which we denote by sk_{Bob} and pk_{Bob} respectively.

- Encryption: Using Bob's public key pk_{Bob}, the sender Alice encrypts her message M, which we call a "plaintext", and obtains a ciphertext C.
- Decryption: Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key sk_{Bob} to recover the plaintext M.

Figure 2.1 illustrates a schematic outline of a public key encryption scheme.

Of course, Bob's public key pk_{Bob} should not compromise the secrecy of the private key sk_{Bob}. This important property is guaranteed by the *trapdoor one-way function* which can be informally defined as follows [78].

> **Trapdoor One-Way Function**: A trapdoor one-way function $f_t(x) : \mathcal{X} \to \mathcal{Y}$ is a one-way function, that is, it is easy to compute $f_t(x)$ for all $x \in \mathcal{X}$ but difficult to invert for almost all values in $\mathcal{Y}$. However, if the trapdoor information t is used, then for all values $y \in \mathcal{Y}$ it is easy to compute $x \in \mathcal{X}$ such that $y = f_t(x)$.

In their seminal paper [45], Diffie and Hellman constructed a trapdoor one-way function based on modulo exponentiation. This function made it possible for them to design a surprising protocol in which the remote users who have not met each other before can share the common secret key. This protocol is now known as the "Diffie-Hellman key exchange protocol".

However, the first practical realization of public key encryption was accomplished by Rivest, Shamir, and Adleman [103] in 1978. Their public key encryption scheme, which we will simply call "`RSA` encryption", can be described as follows:

- Key Generation: The receiver Bob chooses large primes p and q at random; computes $N = pq$; computes $\phi(N) = (p-1)(q-1)$; chooses a random integer $e < \phi(N)$ such that $\gcd(e, \phi(N)) = 1$; computes the integer d such that $ed = 1 \bmod \phi(N)$; publicizes his public key $pk_{Bob} = (N, e)$ while keeps his private key $sk_{Bob} = (p, q, d)$ secret.
- Encryption: Using Bob's public key pk_{Bob}, the sender Alice encrypts her message $M < N$ by creating a ciphertext C such that

$$C = M^e \bmod N.$$

- Decryption: Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key sk_{Bob} and recovers the plaintext M by computing

$$M = C^d \bmod N.$$

Notice that the one-wayness of the above RSA encryption scheme is based on the intractability of computing the e-th root of a ciphertext C modulo integer N, which is related to the "Integer Factorization" problem.

Soon after Rivest, Shamir, and Adleman proposed the above encryption scheme, ElGamal [49] constructed a new public key encryption scheme based on Diffie and Hellman's trapdoor one-way function proposed in [45].

2.2.2 Digital Signature

Another fundamental public key cryptographic scheme is a digital signature, whose concept was first proposed by Diffie and Hellman [45]. As mentioned earlier, the ability to construct a digital signature scheme is a great advantage of public key cryptography over symmetric key cryptography. A digital signature scheme can be described as follows.

- Key Generation: The signer Alice creates her private and public key pair, which we denote by sk_{Alice} and pk_{Alice} respectively.
- Signature Generation: Using her private key sk_{Alice}, Alice creates a signature σ on her message M.
- Signature Verification: Having obtained the signature σ and the message M from Alice, the verifier Bob checks whether σ is a genuine signature on M using Alice's public key pk_{Alice}. If it is, he returns "*Accept*". Otherwise, he returns "*Reject*".

Figure 2.2 illustrates a schematic outline of a digital signature scheme.

Since only a single entity is able to sign a message and the resulting signature is verified by anybody in digital signature, a dispute over who created the signature can be easily settled. This, often called "non-repudiation", is one of the important security services that digital signature schemes can provide. Indeed, non-repudiation is an essential security requirement in electronic commerce applications.

In the same paper that proposed the RSA encryption scheme [103], Rivest, Shamir, and Adleman also constructed a signature scheme, which we call "RSA signature". Below, we describe the RSA signature scheme.

- Key Generation: The signer Alice chooses large primes p and q at random; computes $N = pq$; computes $\phi(N) = (p-1)(q-1)$; chooses a random

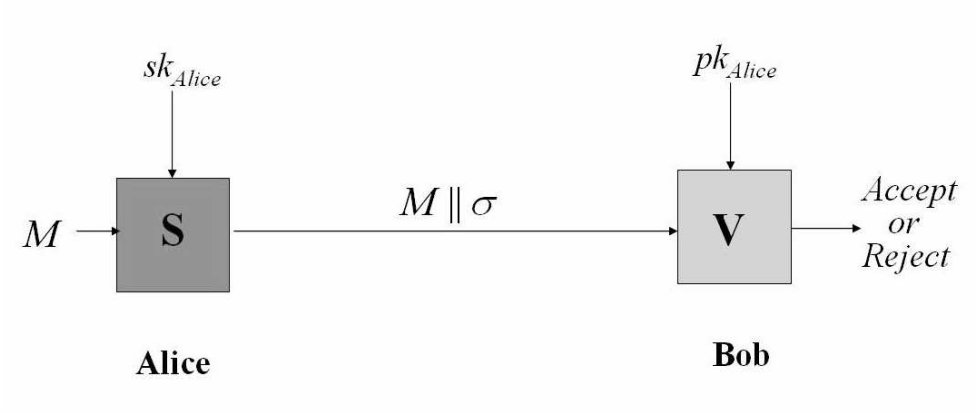


Figure 2.2: Digital Signature

integer $e < \phi(N)$ such that $\gcd(e, \phi(N)) = 1$; computes the integer d such that $ed = 1 \bmod \phi(N)$; publicizes her public key $pk_{Alice} = (N, e)$ while keeps her private key $sk_{Alice} = (p, q, d)$ secret.

- Signature Generation: Using her private key sk_{Alice}, Alice creates a signature on her message $M < N$ by computing

$$\sigma = M^d \bmod N.$$

- Signature Verification: Having obtained the signature σ and the message M from Alice, Bob checks whether

$$M = \sigma^e \bmod N$$

using Alice's public key pk_{Alice}.

If the above equation holds, Bob returns "*Accept*". Otherwise, he returns "*Reject*".

Notice that the unforgeability of the above RSA signature scheme is again based on the intractability of computing the e-th root of a ciphertext C modulo integer N. (However, the "unforgeability" here refers to a weak sense of unforgeability. This issue will be discussed in more detail in Section 2.4.3.)

We remark that the construction of a signature scheme based on the "Discrete-Logarithm" problem, which we will be looking into in Section 2.3.2, was given by ElGamal [49].

2.2.3 Identity-Based Encryption/Signature

The concept of identity-based encryption and signature was originally proposed by Shamir [108] in 1984. In identity-based encryption or signature, one can use the receiver or sender's identifier information such as email or IP address (instead of digital certificates) to encrypt a message or verify a signature. As a result, identity-based cryptographic schemes significantly reduce the system complexity and the cost for establishing and managing the public key authentication framework known as Public Key Infrastructure (PKI).

More precisely, an identity-based encryption scheme can be described using the following steps.

- Setup: The Private Key Generator (PKG), which is a trusted third party, creates its master (private) and public key pair, which we denote by sk_{PKG} and pk_{PKG} respectively. (Note that pk_{PKG} is given to all the interested parties and remains as a constant system parameter for a long period.)
- Private Key Extraction: The receiver Bob authenticates himself to the PKG and obtains a private key $sk_{\mathtt{ID}_{Bob}}$ associated with his identity $\mathtt{ID}_{Bob}$.
- Encryption: Using Bob's identity $\mathtt{ID}_{Bob}$ and the PKG's pk_{PKG}, the sender Alice encrypts her plaintext message M and obtains a ciphertext C.
- Decryption: Upon receiving the ciphertext C from Alice, Bob decrypts it using his private key $sk_{\mathtt{ID}_{Bob}}$ to recover the plaintext M.

Figure 2.3 illustrates a schematic outline of an identity-based encryption scheme.

As a mirror image of the above identity-based encryption, identity-based signature can be described as follows.

- Setup: The Private Key Generator (PKG), which is a trusted third party, creates its master (private) and public key pair, which we denote by sk_{PKG} and pk_{PKG} respectively.
- Private Key Extraction: The sender Alice authenticates herself to the PKG and obtains a private key $sk_{\mathtt{ID}_{Alice}}$ associated with her identity $\mathtt{ID}_{Alice}$.
- Signature Generation: Using her private key $sk_{\mathtt{ID}_{Alice}}$, Alice creates a signature σ on her message M.

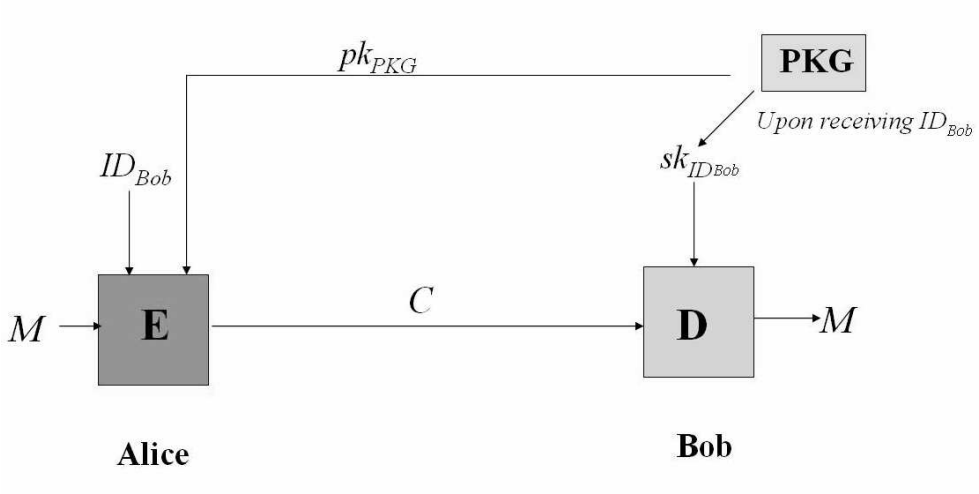


Figure 2.3: Identity-Based Encryption

- Signature Verification: Having obtained the signature σ and the message M from Alice, the verifier Bob checks whether σ is a genuine signature on M using Alice's identity $\mathtt{ID}_{Alice}$ and the PKG's public key pk_{PKG}. If it is, he returns "*Accept*". Otherwise, he returns "*Reject*".

In his original paper on identity-based cryptography [108], Shamir constructed the first identity-based signature scheme using the RSA function. However, he was unable to design an identity-based encryption scheme and posed constructing it as an open problem. Only recently, Shamir's problem was solved by Boneh and Franklin [27] who used the special properties of elliptic curves called the "bilinear pairings" to construct such a scheme. (Boneh and Franklin's scheme will be discussed in great detail in Chapter 5.) Thanks to their successful realization of identity-based encryption, identity-based cryptography is now flourishing in the cryptographic research.

We remark that the PKG in identity-based cryptography must be trusted unconditionally due to the fact that the PKG is in possession of the users' private keys. As a result, the use of identity-based cryptography may be limited to the environment where the PKG is allowed to view the communicated messages in certain circumstances, e.g., a company environment. To resolve this problem, Boneh and Franklin [27] suggested that a master key of the PKG should be distributed into a number of other PKGs called "distributed PKGs". Note that Boneh and Franklin's distributed PKGs technique will be extensively discussed in Chapter 5.

Apart from the very basic features of identity-based cryptography reviewed in this subsection, there are a number of interesting problems to be explored in the field of identity-based cryptography. Readers are referred to Chapter 7 of this thesis and Chapter 13 of Mao's book [78].

2.3 Computational Primitives

The security of a public key cryptographic scheme is relied on the hardness of a certain computational problem. Although various computational problems have been proposed so far, e.g., the braids [4] used in Ko et al.'s encryption scheme [72] and the decoding linear (binary) error-correcting code problem used in the McEliece encryption scheme [83], we only review the "Integer Factorization" and "Discrete-Logarithm" problems, which are the most widely-used computational problems in the conventional cryptographic schemes.

2.3.1 Integer Factorization

The Integer Factorization problem, which we will simply call the "IF problem", can be informally defined as follows:

> **IF**: Given $N = pq$ where p and q are large primes, find p and q.

Recall that the one-wayness of the `RSA` encryption scheme described in Section 2.2.1 and the unforgeability of the `RSA` signature scheme described in Section 2.2.2 lie on the intractability of computing the e-th root of a ciphertext C modulo integer $N = pq$, which is *related* to the above IF problem. The reason why we put the word "related" here is that equivalence between computing e-th root of an element modulo integer N and solving the IF problem has not been proven until now. What we just know is that if the IF problem can be easily (namely, "in polynomial time") solved, then computing $d = e^{-1} \bmod \phi(N)$ can be done also easily since $\phi(N) = (p-1)(q-1)$ can be efficiently computed if p and q are known. On the other had, no answer has been given yet for the question "whether one must factor N to compute $d = e^{-1} \bmod \phi(N)$". For this reason, the computational problem of the `RSA` encryption and signature schemes is specifically called the "RSA problem".

Indeed, factoring a large integer is one of the fascinating subjects in computational number theory. The "quadratic sieve", "elliptic curve", and "number field sieve" methods developed by computational number theorists are currently used

as factoring algorithms in practice. Among them, the number field sieve method, which has been proposed most recently, is especially focused by many cryptographers due to its asymptotic running time faster than those of the quadratic sieve and elliptic curve methods, and its effectiveness to parallelization as shown in factoring a 512-bit RSA modulus [32].

We remark that the IF and its related computational problems were used to build up various cryptographic schemes by Rivest et al. [103], Rabin [99], Okamoto and Uchiyama [89], Pointcheval [96], and Paillier [92].

2.3.2 Discrete-Logarithm

Another widely-used computational problem, the Discrete-Logarithm problem, which we will simply call the "DL problem", can be informally defined as follows:

> **DL**: Given a finite cyclic group $\mathcal{G} = \{g^0, g^1, g^2, \ldots, g^{m-1}\}$ where $m = |\mathcal{G}|$ is the order of $\mathcal{G}$, and a random element $r \in \mathcal{G}$, find the unique integer $i \in \mathbb{Z}_m$ such that such that $r = g^i$.

Motivated by the DL problem, Diffie and Hellman [45] designed the following intriguing key exchange protocol: Alice and Bob share the common cyclic group $\mathcal{G}$, where $|\mathcal{G}| = m$, and its generator g. Alice then chooses a uniformly at random from $\mathbb{Z}_m$, computes g^a, and sends this to Bob. Similarly, Bob chooses b uniformly at random from $\mathbb{Z}_m$, computes g^b, and sends this to Alice. Since Alice and Bob have their private key a and b respectively, they can calculate the same (secret) key g^{ab}.

Note, however, that the attacker for the above key exchange protocol possesses two elements in the group $\mathcal{G}$, that is, g^a and g^b. Since the attacker has the one more additional information g^b, the security of the Diffie-Hellman key exchange protocol is not equivalent to the hardness of the DL problem. For this reason similar to the case of the RSA problem, the computational problem used in the Diffie-Hellman key exchange protocol is specifically called the "Computational Diffie-Hellman (CDH) problem", which can be described as follows:

> **CDH**: Given a finite cyclic group $\mathcal{G} = \{g^0, g^1, g^2, \ldots, g^{m-1}\}$ where $m = |\mathcal{G}|$ is the order of $\mathcal{G}$, and $g^a \in \mathcal{G}$ and $g^b \in \mathcal{G}$ for random $a, b \in \mathbb{Z}_m$, computes $g^{ab} \in \mathcal{G}$.

After the emergence of the CDH problem, researchers realized that the CDH problem by itself is not sufficient for many cryptographic schemes to be proven

secure in that the attacker may have a chance to obtain some valuable information about the key g^{ab} even though its entire part cannot be revealed. This is the motivation for defining the following "Decisional Diffie-Hellman (DDH)" problem:

> **DDH**: Given a finite cyclic group $\mathcal{G} = \{g^0, g^1, g^2, \ldots, g^{m-1}\}$ where $m = |\mathcal{G}|$ is the order of $\mathcal{G}$, and $g^a \in \mathcal{G}$, $g^b \in \mathcal{G}$, and $g^c \in \mathcal{G}$ for random $a, b, c \in \mathbb{Z}_m$, decide whether $c = ab \in \mathbb{Z}_m$.

Note that if one can solve the DL problem easily, he can easily solve the CDH problem too. Likewise, if one can solve the CDH problem easily, he can easily solve the DDH problem. However, the reverse of this reasoning does not generally hold. Readers are referred to [80] and [25] for more detailed discussions on this issue.

Note also that in practice, the group $\mathcal{G}$ can be implemented using the subgroup of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ of order q such that $p = aq + 1$ for primes p and $q > p^{1/10}$, or the group of points on certain elliptic curves of order q [73, 82] for efficiency.

As attack algorithms for the DL problem in a general group $\mathcal{G}$ (e.g., regardless of whether $\mathcal{G}$ is the subgroup of $\mathbb{Z}_p^*$ or the group of points on elliptic curves), Shank's "baby-step, giant step" and Pollard's "rho" methods are used. As a subexponential algorithm for solving the DL problem in $\mathbb{Z}_p^*$, the "index calculus" method is well known.

Finally, we remark that there have been a large number of cryptographic schemes based on the above problems. Examples include the digital signature schemes based on the DL problem such as ElGamal [49], Schnorr [104], and Digital Signature Standard (DSS) [86]; the public key encryption schemes based on the CDH problem such as Pointcheval [95] and Baek-Lee-Kim [5]; the public key encryption scheme based on the DDH problem such as ElGamal [49], Tsiounis-Yung [121], and Cramer-Shoup [41].

2.4 Security Notions

2.4.1 Steps to Achieving Provable Security

As discussed in Chapter 1, provable security evaluates the security of a given cryptographic scheme by presenting a reduction between the properly defined security notion for the scheme and the underlying primitive which is known to be secure. (In fact, the concept of a reduction is originally from the theory

of computation. Informally speaking, it is a way of converting one problem to another problem in such a way that a solution to the latter problem can be used to solve the former one [114].) In [14], Bellare explains precisely how to achieve provable security, which can be summarized as the following steps:

1. Set up a security goal, e.g. confidentiality via encryption or authenticity via signature;
2. Construct a formal attack model and define what it means for a cryptographic scheme to be secure;
3. Show by a reduction that the only way to break the security of cryptographic schemes is to solve computationally hard problems or break other primitives.

Actually, setting up security goals and constructing relevant attack models, in other words, formulating right definitions for the security of cryptographic schemes is important by itself. Over the last two decades, researchers have been proposing various security notions (definitions of security) for cryptographic schemes either in public key or symmetric key setting. In the next two subsections, we survey some important confidentiality and unforgeability notions widely used in public key cryptography.

2.4.2 Confidentiality Notion

To begin with, let us recall that the `RSA` encryption scheme described in Section 2.2.1 is secure in the "one-wayness" sense: Unless the attacker Marvin obtains Bob's private key, he cannot recover the whole part of plaintext.

However, we are not too sure whether the `RSA` encryption scheme will still be secure in situations where something more than one-wayness is required. In fact, the `RSA` encryption scheme is not secure in the following situation: Assume that members of a committee use a confidential on-line poll to decide on some course of action and the `RSA` encryption scheme is employed to encrypt the members' votes. But, there is an additional assumption that the committee members should use one of the predetermined messages, "Yes" and "No", to create their ciphertext. After encrypting one of those messages, each of the members sends over the ciphertext to a chairperson, who is not supposed to know who votes for "Yes" or "No". However, the corrupted chairperson will know who has voted for which by

re-encrypting the guessed plaintext ("Yes" or "No") to see if the resulting ciphertext matches the one in his hand since there are only two forms of ciphertexts C_{Yes} and C_{No} which encrypt "Yes" and "No" respectively.

The above problem was actually the main motivation for Goldwasser and Micali's [62] notion of confidentiality for public key encryption called "semantic security ", equivalently known as "(polynomial) indistinguishability of encryptions under chosen-plaintext attack (IND-CPA) [16]". The implication of this notion is that a ciphertext should not reveal any partial information about the plaintext apart from its length to any attacker whose computational power is polynomially bounded.

Yet, one can imagine even a harsher situation where the attacker Marvin may be provided with a decryption oracle (algorithm), from which he gets decryptions of some ciphertexts of his choice even if he is not in possession of the decryption key. This is so-called "chosen ciphertext attack", appeared and evolved in the series of papers by Naor and Yung [85], Rackhoff and Simon [100], and Dolev, Dwork, and Naor [48]. When conducting chosen ciphertext attack, Marvin's chance to obtain useful information about a plaintext given its encryption which is called a "target ciphertext" can be very high in some schemes. (Of course, it is assumed that the attacker cannot get a decryption of a target ciphertext.)

In fact, the `RSA` encryption scheme described in Section 2.2.1 is very weak under this attack as demonstrated in the following: Assume that Bob's public and private keys are (N, e) and (p, q, d) such that $N = pq$, where p and q are large primes chosen at random, and $ed = 1 \bmod \phi(N)$ where $\phi(N) = (p-1)(q-1)$. Suppose that Marvin has obtained a target ciphertext $C = M^e \bmod N$ which encrypts M. Now, Marvin just chooses an arbitrary message $R \in N$, computes $C' = R^e C$, and queries it to the decryption oracle (algorithm) and gets $M' = C'^d$. Marvin then computes M'/R. Since

$$M' = C'^d = (R^e C)^d = RC^d = RM,$$

M'/R is the message M which is the plaintext of C!

As seen from the above two attacks, the `RSA` encryption scheme in Section 2.2.1 as it is cannot be used on its own. In practice, RSA-OAEP, a converted version of the `RSA` encryption scheme using Bellare and Rogaway [18]'s "Optimal Asymmetric Encryption Padding (OAEP)", is widely used. Indeed, OAEP fixes the above two security problems of the `RSA` encryption scheme. More precisely, it makes the `RSA` encryption scheme *provably secure* in the "indistinguishability of encryptions under chosen ciphertext attack (IND-CCA)" sense under the random

oracle assumption which will be discussed in detail in Section 2.5. (We remark, however, that the original proof given in [18] that OAEP applies to *any* one-way trapdoor permutation was later found to be false by Shoup [110]. Shortly after Shoup's work, Fujisaki, Okamoto, Pointcheval, and Stern [56] confirmed that OAEP in fact applies to the *sole* RSA function.)

Now, we look into the IND-CCA notion in the context of modern cryptography. In the current literature, the IND-CCA notion is usually described in terms of the following game in which the attacker and the "Challenger" interact each other:

> **Phase 1**: The Challenger generates a private/public key pair and all the necessary common parameters of a given public key encryption scheme in the prescribed manner. The Challenger then gives the public key and the common parameters to the attacker while keeps the private key secret.
>
> **Phase 2**: The attacker queries a number of ciphertexts to the Challenger to obtain their decryptions. Upon receiving each of the ciphertexts, the Challenger decrypts it using the private key he generated in Phase 1 and returns the resulting decryption to the attacker.
>
> **Phase 3**: The attacker chooses two equal-length plaintexts (M_0, M_1) and gives them to the Challenger. Upon receiving M_0 and M_1, the Challenger chooses one of them at random and computes its encryption. The Challenger returns the resulting ciphertext (called a "target ciphertext") to the attacker.
>
> **Phase 4**: The attacker again queries a number of ciphertexts to the Challenger to obtain their decryptions, subject to the *restriction* that the attacker cannot query the (target) ciphertext obtained in Phase 3. Upon receiving each of the ciphertexts, the Challenger decrypts it using the private key he generated in Phase 1 and returns the result (decryption) to the attacker.
>
> **Phase 5**: Finally, the attacker outputs his guess on which one of M_0 and M_1 was chosen by the Challenger in Phase 3.

If no attacker can guess correctly with probability significantly greater than $1/2$ in Phase 5, the given public key encryption scheme is said to be secure in the IND-CCA sense.

We remark that it can be shown that the IND-CCA notion implies the IND-CPA notion, that is, every public key encryption scheme meeting the IND-CCA

notion also meets the IND-CPA notion. Hence, obtaining a public key encryption scheme secure in the IND-CCA sense means that we have already obtained an IND-CPA secure one. (Relationships among various notions of confidentiality of public key encryption are formally discussed in [16].)

We also remark that the chosen ciphertext attack discussed in this section is sometimes referred to in the literature as the "*adaptive* chosen ciphertext attack" to emphasize that the attacker is allowed to query ciphertexts to decryption oracle in an adaptive way before and *after* he obtains a target ciphertext. For this reason, the IND-CCA notion described above is sometimes referred to as "IND-CCA2" [16]. However, throughout this thesis, we simply use "chosen ciphertext attack" and "IND-CCA" to refer to "adaptive chosen ciphertext attack" and "IND-CCA2" respectively.

A final remark is that the above IND-CCA notion can be extended to the chosen ciphertext security notion for identity-based encryption, called "IND-ID-CCA" [27]. In this notion, the attacker is not only allowed to make decryption queries, but also to make a number of private key extraction queries to the PKG to obtain private keys corresponding to some identities of his choice. An encryption scheme should remain secure under this attack to be IND-ID-CCA secure.

2.4.3 Unforgeability Notion

Care should be also taken to evaluate the security of a digital signature scheme since the attacker can mount "chosen message attack". In this attack, the forger Marvin has access to Alice's signature generation oracle (algorithm) from which Marvin gets signatures of any messages of his choice. At the end of the attack, he returns a *new* message-signature pair as a forgery. (That is, the message has not been queried to the signature generation oracle before.) Note that this type of forgery is called the "existential forgery".

Indeed, the `RSA` signature scheme described in Section 2.2.2 is existentially forgeable under chosen message attack: Assume that Alice's public and private keys are (N, e) and (p, q, d) such that $N = pq$, where p and q are large primes chosen at random, $ed = 1 \bmod \phi(N)$ where $\phi(N) = (p-1)(q-1)$. Now, Marvin queries chooses two messages $M_1 \in N$ and $M_2 \in N$ and gets signatures $\sigma_1 = M_1^d$ and $\sigma_2 = M_2^d$ from the signature generation oracle. Marvin then computes $M' = M_1 M_2$ and $\sigma' = \sigma_1 \sigma_2$, and outputs (M', σ') as a forgery. Since

$$\sigma'^e = (\sigma_1 \sigma_2)^e = (M_1^d M_2^d)^e = M_1 M_2,$$

σ' is a valid signature for the message M'!

One may, however, claim that the existential forgery is a too strong security requirement since the output message is likely to be meaningless. However, in the current computing environment, not only meaningful messages but also arbitrary bit streams such as keys, image files, program code [94] can be signed, so the existential forgery can cause serious damage.

We remark that in order to prevent the above attack on the `RSA` signature scheme, we need a hash function to digest the message before it is signed. In [19], it was shown that if the hash function is assumed to be a random oracle, the `RSA` signature scheme can be "existentially unforgeable under chosen message attack (UF-CMA)".

Now, we look into the UF-CMA notion in the context of modern cryptography as we previously did for the IND-CCA notion. The UF-CMA notion was first formalized by Goldwasser, Micali, and Rivest [63] and is usually described in terms of the following attack game in which the attacker and the "Challenger" interact each other:

> **Phase 1**: The Challenger generates a private/public key pair and all the necessary common parameters of a given digital signature scheme in the prescribed manner. The Challenger then gives the public key and the common parameters to the attacker while keeps the private key secret.
>
> **Phase 2**: The attacker queries a number of messages to the Challenger to obtain signatures on them. Upon receiving each of the messages, the Challenger generates a signature using the private key he generated in Phase 1 and returns the resulting signature to the attacker.
>
> **Phase 3**: The attacker outputs a new message-signature pair. (Note that the message has not been queried to the Challenge for signature generation in Phase 2.)

The signature scheme is said to be secure in the UF-CMA sense if no attacker can succeed in Phase 3 with great probability.

Similarly to the case of IND-CCA, the above UF-CMA can be extended to the unforgeability notion for identity-based signature, which we call "UF-IDS-CMA" [34, 66]. In this notion, the attacker is not only allowed to make signature generation queries, but also to make a number of private key extraction queries to the PKG to obtain private keys corresponding to some identities of his choice. If

a signature scheme remains secure under this attack, it is said to UF-IDS-CMA secure. (Note that the UF-IDS-CMA will be discussed in detail in Chapter 6.)

The IND-CCA and UF-CMA notions rather informally introduced in the previous and this subsections serve as fundamental security notions in public key cryptography. These notions will be used directly or further evolved to analyze the various cryptographic schemes which will be presented in the rest of the chapters. Before moving to the next chapter, we talk about the random oracle model, which is a controversial but useful tool for designing efficient and provably secure cryptographic schemes.

2.5 Random Oracle Model

After formulating security notions, the next step one should take in the provable security approach is to show by a reduction that the only way to break the security of a given cryptographic scheme is to solve a related computationally hard problem. However, this is not always easy unless hash functions used in the construction of cryptographic scheme are assumed to behave as completely random functions.

The random oracle model, first appeared in [51] and popularized by Bellare and Rogaway [20], gives a mathematical model of such ideal hash functions. In this model, a hash function $h : \mathcal{X} \to \mathcal{Y}$ is chosen at random from $S^{\mathcal{X},\mathcal{Y}}$ which denotes the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, and evaluation of h on inputs in $\mathcal{X}$ can be done only by querying the random oracle. According to Anderson [3], the random oracle can be compared with a black box in which an elf is sitting with a source of randomness and some means of storage, which are represented as a dice and a scroll respectively. The elf accepts inputs of a certain type from the outside, then looks up the scroll to see whether this query has been answered before. If so, the elf returns the corresponding answer again; otherwise, he randomly generates an answer by throwing the dice.

Hence, by the assumptions made in the random oracle model, we obtain the following key property:

- Assume that $h \in S^{\mathcal{X},\mathcal{Y}}$ is chosen at random. Fix $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Then we have $\Pr[h(x) = y] = 1/|\mathcal{Y}|$.

However, a problem of the random oracle model is that the behavior of the random oracles is so ideal that no realization is possible. What one can do is to replace the random oracles by the conventional hash functions such as SHA-1 [87]

or MD5 [102] when the cryptographic schemes that use them are implemented. For this reason, the use of the random oracle model is somewhat controversial. Canetti, Goldreich, and Halevi were even able to demonstrate that there exist some special signature and encryption schemes which are secure in the random oracle model but become insecure whenever the random oracles are specified [30]. However, there is also a strong trend that security proofs in the random oracle at least give a certain level of security guarantees although not at the same level as those of the standard provable security approach, and more importantly, the schemes designed in the random oracle are usually very efficient [14]. Actually, it is becoming a consensus that today's standards should include the schemes with proof in the random oracle model rather than those without.

In this thesis, we take this latter view on the random oracle model. So the schemes presented in the rest of chapters are supported by proofs in the random oracle model. However, we do support the criticism on the random oracle model and agree with the opinion that more research should be done on the realization of the random oracles or moving towards a more realistic model.

Chapter 3

Security Analysis of Zheng and Seberry's Public Key Encryption Scheme

3.1 Introduction

3.1.1 Motivation

Since Diffie and Hellman [45] proposed the concept of public key encryption, design of secure yet efficient public key encryption schemes has been a challenging task for many cryptographers. In addition to security and efficiency, simplicity of public key encryption schemes has been regarded as of particular importance due to the fact that a number of cryptographic software packages are actually implemented by application programmers who are not experts in cryptography.

For these reasons, the three earliest, surprisingly simple and efficient schemes proposed by Zheng and Seberry at Crypto '92 [130] are still worth focusing, although more than 10 years have passed since they were proposed. It is interesting to notice that a number of recently proposed efficient and provably secure public key encryption schemes, including DHIES [1] and REACT [91] bear a close resemblance to one of the schemes proposed by Zheng and Seberry, which we call a "Zheng-Seberry scheme". This particular scheme is the focus of this chapter. In spite of its design simplicity and efficiency, a problem remaining with the Zheng-Seberry scheme is that it could not be analyzed using the *provable security* approach discussed in the previous chapters: As shown by Lim and Lee

[77], known plaintext attack is applicable to the Zheng-Seberry scheme, which in effect makes it impossible to prove that the scheme is secure under the IND-CPA or IND-CCA notion, where two challenge plaintexts are assumed to be known to the attacker. In the same paper, however, Lim and Lee proposed a countermeasure for the attack and this was reflected in the new descriptions of the scheme by Zheng [128]. But, no formal security analysis for this was presented in neither [77] nor [128].

3.1.2 Contributions of This Chapter

The contributions of this chapter are twofold. First, we show that Lim and Lee's modified version of the Zheng-Seberry scheme is indeed provably secure against chosen ciphertext attack in the random oracle model, relative to the Gap Diffie-Hellman problem [90]. Second, we show, through a careful analysis, that the security proof for the Zheng-Seberry scheme recently claimed by Soldera, Seberry and Qu [115] is in fact false.

3.2 Preliminaries

In this section, we review chosen ciphertext security for public key encryption. Compared with Chapter 2, the definitions given in this section are more formal.

3.2.1 Chosen Ciphertext Security Notion for Public Key Encryption

We first review the definition of a public key encryption scheme, which we denote by "$\mathcal{PKE}$".

Definition 1 (Public Key Encryption) A public key encryption scheme $\mathcal{PKE}$ consists of the following algorithms:

- A randomized common parameter generation algorithm $\mathsf{GK}(k)$: Given a security parameter $k \in \mathbb{N}$, this algorithm generates a set of public common parameters, e.g., descriptions of hash functions and a mathematical group. The output of this algorithm denoted by cp includes such parameters as well as the security parameter k.

- A randomized key generation algorithm $\mathsf{GK}(cp)$: Given a security parameter cp, this algorithm generates a key pair (sk, pk) where sk and pk denote the private key and public key respectively.
- A randomized encryption algorithm $\mathsf{E}(cp, pk, m)$: Given a public key pk and a plaintext m, this algorithm generates a ciphertext denoted by c.
- A deterministic decryption algorithm $\mathsf{D}(sk, c)$: Given a private key sk and a ciphertext c, this algorithm outputs a plaintext m or a special symbol "*Reject*".

We now review the IND-CCA (indistinguishability of encryptions under chosen ciphertext attack) notion for public key encryption. (Note that this notion is sometimes referred to as "IND-CCA2" [16].)

Definition 2 (IND-CCA) Let $\mathcal{PKE} = (\mathsf{GC}, \mathsf{GK}, \mathsf{E}, \mathsf{D})$ be a public key encryption scheme. Let $\mathsf{A}^{\mathsf{CCA}}$ be an attacker modelled as a probabilistic Turing machine. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{CCA}}$ interacts with the "Challenger".

Phase 1: Given a security parameter k, the Challenger runs the common parameter generation algorithm $\mathsf{GC}(k)$ and obtains a common parameter cp. The Challenger then runs the key generation algorithm $\mathsf{GK}(cp)$ to generate the receiver's private/public key pair (sk, pk). The Challenger gives cp and pk to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 2: $\mathsf{A}^{\mathsf{CCA}}$ submits a number of queries, each of which consists of a ciphertext c, to the decryption oracle. On receiving c, the Challenger runs $\mathsf{D}(cp, sk, c)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 3: $\mathsf{A}^{\mathsf{CCA}}$ chooses two equal-length plaintexts (m_0, m_1). On receiving these, the Challenger chooses $\beta \in \{0, 1\}$ at random, computes a target ciphertext $c^* = \mathsf{E}(cp, pk, m_\beta)$, and gives it to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 4: $\mathsf{A}^{\mathsf{CCA}}$ continues to submit a number of queries to the decryption oracle. As in Phase 2, each of the queries consists of a ciphertext c. However, a restriction in this phase is that $c \neq c^*$. On receiving c, the Challenger runs $\mathsf{D}(cp, sk, c)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 5: $\mathsf{A}^{\mathsf{CCA}}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$.

We define $\mathsf{A}^{\mathsf{CCA}}$'s success by the probability

$$\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathsf{A}^{\mathsf{CCA}},\mathcal{PKE}}(k) \stackrel{\mathrm{def}}{=} 2\mathrm{Pr}[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE}}(t_{CCA}, q_D)$ the maximum of the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CCA}}$ having running time t_{CCA} and making at most q_D queries to the decryption oracle. Note that the running time and the number of queries are all polynomial in the security parameter k.

The $\mathcal{PKE}$ scheme is said to be secure in the IND-CCA sense if $\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathcal{PKE}}(t_{CCA}, q_D)$ is negligible in k. (Throughput this thesis, a probability function $f : \mathbb{N} \to \mathbb{R}_{[0,1]}$ is said to be *negligible* in k if, for all $c > 0$, there exists $k_0 \in \mathbb{N}$ such that $f(k) \leq \frac{1}{k^c}$ whenever $k \geq k_0$, where $\mathbb{R}_{[0,1]} = \{x \in \mathbb{R} | 0 \leq x \leq 1\}$.)

3.2.2 Diffie-Hellman Primitives

In this section, we review the definitions of the Decisional Diffie-Hellman (DDH) and Gap Diffie-Hellman (GDH) problems, which we will use later in this chapter.

As mentioned in Section 2.3.2, the DDH problem emerged from the difficulty of proving the security of many Diffie-Hellman-based cryptographic schemes. In what follows, we review the formal definition of the DDH problem.

Definition 3 (DDH) Let $\mathcal{G}$ be a cyclic group of order $q \geq 2^k$ generated by $g \in \mathcal{G}$, where q is a prime and k is a security parameter. Let $\mathsf{A}^{\mathsf{DDH}}$ denote an attacker assumed to be a probabilistic Turing machine taking the security parameter k as input. Suppose that a and b are uniformly chosen at random from $\mathbb{Z}_q^*$ and g^a and g^b are computed.

$\mathsf{A}^{\mathsf{DDH}}$ is to solve the following problem:

- Given $(\mathcal{G}, q, g, g^a, g^b, g^c)$, outputs 1 if $c = ab$ and 0 otherwise.

We define the attacker $\mathsf{A}^{\mathsf{DDH}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{DDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{DDH}}}(k) \stackrel{\mathrm{def}}{=} \Big| \mathrm{Pr}[\mathsf{A}^{\mathsf{DDH}}(\mathcal{G}, q, g, g^a, g^b, g^{ab}) = 1] - \mathrm{Pr}[\mathsf{A}^{\mathsf{DDH}}(\mathcal{G}, q, g, g^a, g^b, g^r) = 1] \Big|$$

for random $r \neq ab \in \mathbb{Z}_q$.

We denote by $\mathbf{Succ}^{\mathrm{DDH}}_{\mathcal{G}}(t_{DDH})$ the maximal success probability $\mathbf{Succ}^{\mathrm{DDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{DDH}}}(k)$ over all attackers $\mathsf{A}^{\mathsf{DDH}}$ having running time t_{DDH} which is polynomial in the security parameter k.

The DDH problem is said to be computationally intractable in the group $\mathcal{G}$ if $\mathbf{Succ}^{\mathrm{DDH}}_{\mathcal{G}}(t_{DDH})$ is negligible in k.

In 2001, Okamoto and Pointcheval proposed another variant of the Diffie-Hellman problem, called "Gap Diffie-Hellman (GDH)". In this problem, the attacker is to solve the Computational Diffie-Hellman (CDH) problem with the help of the DDH oracle which tells whether a given tuple is Diffie-Hellman tuple or not. Note that if the DDH oracle helped so much that the attacker can solve the CDH problem, the GDH problem would not be meaningful. However, it has not been proven so far that the DDH problem can be used to solve the CDH problem and hence the GDH problem is a reasonable computational problem.

In fact, the GDH problem has been able to prove various cryptographic schemes which had not been proven before. Examples include the undeniable signature [36], designated confirmer signature [35], and DHIES [1] schemes. Also, the new public key encryption schemes based on this problem have emerged, e.g., REACT [91] and GEM [40]. Below, we review the formal definition of the GDH problem.

Definition 4 (GDH) Let $\mathcal{G}$ be a cyclic group of order $q \geq 2^k$ generated by $g \in \mathcal{G}$, where q is a prime and k is a security parameter. Let $\mathsf{A}^{\mathsf{GDH}}$ denote an attacker assumed to be a probabilistic Turing machine taking the security parameter k as input. Suppose that a and b are uniformly chosen at random from $\mathbb{Z}^*_q$ and g^a and g^b are computed.

$\mathsf{A}^{\mathsf{GDH}}$ is to solve the following problem:

- Given $(\mathcal{G}, q, g, g^a, g^b)$, compute the Diffie-Hellman key g^{ab} of g^a and g^b with the help of the Decisional Diffie-Hellman oracle $\mathcal{O}^{DDH}_g(\cdot,\cdot,\cdot)$, which, given (g^u, g^v, g^w) where g is the (fixed) generator of $\mathcal{G}$, outputs 1 if $w = uv$ and 0 otherwise.

We define $\mathsf{A}^{\mathsf{GDH}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{GDH}}(\mathcal{G}, q, g, g^a, g^b) = g^{ab}].$$

We denote by $\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(t_{GDH}, q_{DDH})$ the maximal success probability $\mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k)$ over all attackers $\mathsf{A}^{\mathsf{GDH}}$ having running time t_{GDH} and the number of queries to

the DDH oracle is bounded by q_{DDH}. The running time t_{DDH} and the number of DDH oracle queries q_{DDH} are polynomial in the security parameter k.

The GDH problem is said to be computationally intractable if $\mathbf{Succ}_{\mathcal{G}}^{\mathrm{GDH}}(t_{GDH}, q_{DDH})$ is negligible in k.

3.3 The Modified Zheng-Seberry Scheme

Recall that in the Zheng-Seberry scheme [130], a plaintext message m is encrypted by creating a ciphertext

$$c = (g^r, G(y^r) \oplus (m||H(m)),$$

where G, H are hash functions and y is a public key such that $y = g^x$ for a private key $x \in \mathbb{Z}_q^*$. Throughout this paper, "||" denotes a concatenation.

The structure of the modified Zheng-Seberry scheme described in [77] is basically the same as the original one described above. The only difference is that in the modified scheme, the Diffie-Hellman key y^r as well as the message m is provided as input to the hash function H. Intuitively, this seems to prevent the known plaintext attack presented by Lim and Lee. Indeed, we show in a later section that this intuition is correct.

3.3.1 Description of the the Modified Zheng-Seberry Scheme

Now, we describe the modified Zheng-Seberry scheme, which we denote by "MZS". Note that in the description below, the plaintext message m is assumed to be drawn from the space $\{0,1\}^{k_0}$ for $k_0 \in \mathbb{N}$.

- A randomized common parameter generation $\mathsf{GC}(k)$
  - Choose a group $\mathcal{G}$ of prime order $q \geq 2^k$.
  - Choose a generator g of $\mathcal{G}$.
  - Find a real constant $\lambda \geq 1$ such that λk is an element of $\mathbb{N}$.
    - Let $k_1 = \lambda k$.
  - Pick a hash function $H : \{0,1\}^{k_0} \times \mathcal{G} \rightarrow \{0,1\}^{k_1}$ modelled as a random oracle.
  - Pick a hash function $G : \mathcal{G} \rightarrow \{0,1\}^{k_0+k_1}$ modelled as a random oracle.

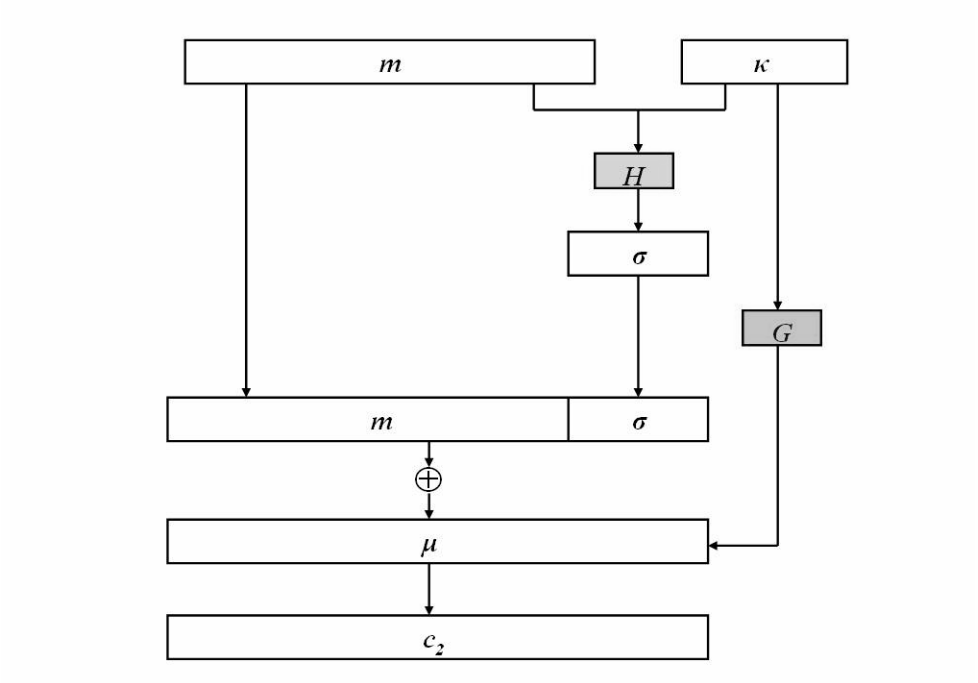


Figure 3.1: Padding Method of the Modified Zheng-Seberry Scheme

  - Output a common parameter $cp = (\mathcal{G}, g, q, G, H, k, k_0, k_1)$.

- A randomized key generation algorithm $\mathsf{GK}(cp)$
  - Pick x uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $y = g^x$.
  - Output a public key $pk = (cp, y)$ and a private key $sk = (cp, x)$.
- A randomized encryption algorithm $\mathsf{E}(pk, m)$
  - Pick r uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $c_1 = g^r$ and $\kappa = y^r$.
  - Compute $\mu = G(\kappa)$ and $\sigma = H(m||\kappa)$.
  - Compute $c_2 = \mu \oplus (m||\sigma)$.
  - Output a ciphertext $c = (c_1, c_2)$.
- A deterministic decryption algorithm $\mathsf{D}(sk, c)$

- Parse c as (c_1, c_2).
- Compute $\kappa = c_1^x$ and $\mu = G(\kappa)$.
- Compute $t = c_2 \oplus \mu$.
  - Let $[t]^{k_0}$ be the first k_0-bits of t, starting with the first bit.
  - Let $[t]_{k_1}$ be the remaining k_1-bits of t, starting with the (k_0+1)-th bit.
- Compute $\sigma = H([t]^{k_0}||\kappa)$.
- If $[t]_{k_1} = \sigma$, do the following:
  - Define m as $[t]^{k_0}$ and output m.
- Else output "*Reject*".

We refer readers to Figure 3.1 which illustrates the padding method of the `MZS` scheme.

3.4 Security Analysis of the Modified Zheng-Seberry Scheme

In this section, we analyze the security of the `MZS` scheme. For careful analysis, we use the proof methodology introduced by Shoup [110]: We start with the real attack game where the attacker $\mathsf{A}^{\mathsf{CCA}}$ is to defeat the security of the Zheng-Seberry scheme in the IND-CCA sense (Definition 2). We then modify this game by changing its rules which describe how variables in the view of $\mathsf{A}^{\mathsf{CCA}}$ are computed, and obtain a new game. We repeat the modification until we simulate the view of $\mathsf{A}^{\mathsf{CCA}}$ completely and obtain a game related to the ability of the attacker $\mathsf{A}^{\mathsf{GDH}}$ to solve the GDH problem (Definition 4). When a new game is derived from a previous one, a difference of the views of the attacker in each game might occur. This difference is measured by the technique presented in the following lemma.

Lemma 1 *Let* A_1, A_2, B_1 *and* B_2 *be events defined over some probability space. If* $\Pr[\mathsf{A}_1 \wedge \neg\mathsf{B}_1] = \Pr[\mathsf{A}_2 \wedge \neg\mathsf{B}_2]$ *and* $\Pr[\mathsf{B}_1] = \Pr[\mathsf{B}_2] = \varepsilon$ *then we have* $|\Pr[\mathsf{A}_1] - \Pr[\mathsf{A}_2]| \leq \varepsilon$.

The proof is a straightforward calculation and can be found in [110]. Now, we state and prove the following theorem.

Theorem 1 *The modified Zheng-Seberry scheme* MZS *is IND-CCA secure in the random oracle model, assuming the GDH problem in group* $\mathcal{G}$ *is computationally intractable. More precisely, we obtain the following bound:*

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{MZS}}(t_{CCA}, q_G, q_H, q_D) \leq \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(t_{GDH}, q_{DDH}) + \frac{q_D}{2^{k_1-1}},$$

where $t_{GDH} = t_{CCA} + q_G + q_H + q_D(q_G + q_H)(T_{DDH} + O(1))$ *and* $q_{DDH} \leq q_G + q_H + q_D(q_G + q_H)$.

Note that q_G, q_H *and* q_D *denote the number of queries made by the IND-CCA attacker having running time* t_{CCA} *to the random oracles* G *and* H, *and the decryption oracle respectively. Note also that* q_{DDH} *denotes the number of queries made by the GDH attacker having running time* t_{GDH} *to the DDH oracle* $\mathcal{O}^{DDH}_g$ *whose running time is bounded by* T_{DDH}.

Proof. Let $\mathsf{A}^{\mathsf{CCA}}$ be an IND-CCA attacker whose running time is polynomial in a security parameter k. Also, let $\mathsf{A}^{\mathsf{GDH}}$ be an attacker trying to solve the GDH problem, given $(\mathcal{G}, q, g, g^a, g^b)$.

As mentioned earlier, we start with the following game which is equivalent to the real attack game.

- Game $\mathbf{G_0}$: This game is actually the same as the real attack game described in Definition 2.

  First, we run the common parameter generation algorithm of the MZS scheme taking the security parameter k as input and obtain a common parameter cp. Then, we run the key generation algorithm on input cp and obtain a private key (cp, x) and a public key (cp, y), where $y = g^x$. The public key $pk \stackrel{\text{def}}{=} (cp, y)$ is given to $\mathsf{A}^{\mathsf{CCA}}$. After $\mathsf{A}^{\mathsf{CCA}}$ submits a pair of plaintexts (m_0, m_1), we create a target ciphertext $c^* = (c_1^*, c_2^*)$ as follows.

  $$c_1^* = g^{r^*}; c_2^* = G(\kappa^*) \oplus (m_\beta || \sigma^*),$$

  where

  $$\kappa^* = y^{r^*}; \sigma^* = H(m_\beta || \kappa^*)$$

for r^* and β picked uniformly at random from $\mathbb{Z}_q^*$ and $\{0,1\}$ respectively. On input c^*, $\mathsf{A}^{\mathsf{CCA}}$ outputs $\beta' \in \{0,1\}$. We denote by S_0 the event $\beta' = \beta$ and use a similar notation S_i for all $\mathbf{G}_i$.

Since this game is the same as the real attack game, we have

$$\Pr[S_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{MZS},\mathsf{A}^{\mathsf{CCA}}}(k).$$

- Game $\mathbf{G_1}$: In this game, we modify the encryption oracle (creation of a target ciphertext) presented in the previous game. Our modification obeys the following rules.

  R1-1 First, we choose $\kappa^+ \in \mathcal{G}$, $c_1^+ \in \mathcal{G}$, $\mu^+ \in \{0,1\}^{k_0+k_1}$ and $\sigma^+ \in \{0,1\}^{k_1}$ uniformly at random. Then, we replace c_1^*, κ^*, $G(\kappa^*)$ and $H(m_\beta||\kappa^*)$ in the target ciphertext c^* by c_1^+, κ^+, μ^+ and σ^+ respectively. Accordingly, we replace c_2^* in c^* by $c_2^+ = \mu^+ \oplus (m_\beta||\sigma^+)$. A new target ciphertext is (c_1^+, c_2^+) and is denoted by c_+^*.

  R1-2 Whenever the random oracle G is queried at κ^+, we respond to it with μ^+.

  R1-3 Whenever the random oracle H is queried at $(m||\kappa^+)$ for some $m \in \{0,1\}^{k_0}$, we respond to it with σ^+.

  The attacker $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_0}$ and Game $\mathbf{G_1}$ since we have replaced one set of random variables by another set of random variables which is different, yet has the same distribution.

  In this game, we assume that the decryption oracle is perfect. That is, receiving $\mathsf{A}^{\mathsf{CCA}}$'s decryption query, $c = (c_1, c_2)$ which is different from the target ciphertext, we decrypt it in the same way as we do in the real attack game. We refer to this rule as "**R1-4**".

  Accordingly, we have

  $$\Pr[S_1] = \Pr[S_0].$$

- Game $\mathbf{G_2}$: In this game, we retain the rules **R1-1** and **R1-4**, renaming them as "**R2-1**" and "**R2-4**" respectively. However, we drop the rules **R1-2** and **R1-3**. That is, μ^+ and σ^+ are used only in the encryption oracle for producing the target ciphertext c_+^* while in other cases when the decryption oracle queries to the random oracles G and H, or $\mathsf{A}^{\mathsf{CCA}}$ directly queries to

them, answers from G or H are taken. We refer to these rules regarding the random oracles G and H as "**R2-2**" and "**R2-3**" respectively.

Since we have dropped the rule **R1-2**, the input to $\mathsf{A}^{\mathsf{CCA}}$ follows a distribution that does not depend on β. Hence, we get $\Pr[S_2] = 1/2$.

Also, note that Game $\mathbf{G_1}$ and Game $\mathbf{G_2}$ may differ if G is queried at κ^+ or H is queried at $(m||\kappa^+)$ for some $m \in \{0,1\}^{k_0}$. Let $\mathsf{AskG}_2 \vee \mathsf{AskH}_2$ denote an event that, in Game $\mathbf{G_2}$, G is queried at κ^+ or H is queried at $(m||\kappa^+)$. For notational convenience, let $\mathsf{AskKey}_2 = \mathsf{AskG}_2 \vee \mathsf{AskH}_2$. We will use an identical notation AskKey_i for all the remaining games.

Now, we have

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{AskKey}_2].$$

- Game $\mathbf{G_3}$: In this game, we again modify the target ciphertext $c^*_+ = (c_1^+, c_2^+)$, where $c_2^+ = \mu^+ \oplus (m_\beta||\sigma^+)$ and $\mu^+ = G(\kappa^+)$, produced by the rule **R2-1** in Game $\mathbf{G_2}$. We replace this rule by the following new rule **R3-1**.

  > **R3-1** First, we replace y the public key element given to $\mathsf{A}^{\mathsf{CCA}}$ by g^b, and c_1^+ by g^a, where (g^a, g^b) are the parameters given to the attacker $\mathsf{A}^{\mathsf{GDH}}$. Now, we define κ^+ as y^a, μ^+ as $G(y^a)$ and σ^+ as $H(m_\beta||y^a)$. That is, we use the same c_2^+ in c^*_+ for the second element of a new target ciphertext c^*_{DH} but we rename the values in it. Therefore, the resulting target ciphertext denoted by c^*_{DH} is (g^a, c_2^+), where $c_2^+ \stackrel{\mathrm{def}}{=} G(y^a) \oplus (m_\beta||H(m_\beta||y^a))$.

  However, we retain the rules **R2-2**, **R2-3** and **R2-4** in Game $\mathbf{G_2}$, renaming them as "**R3-2**", "**R3-3**" and "**R3-4**" respectively.

  Thanks to the randomness of the oracle G and the uniformity of g^a in the group $\mathcal{G}$, $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_2}$ and Game $\mathbf{G_3}$. Hence, we have

  $$\Pr[\mathsf{AskKey}_3] = \Pr[\mathsf{AskKey}_2].$$

  From now on, we deal with the decryption oracle which has been regarded as perfect up to this game.

- Game $\mathbf{G_4}$: We retain all the rules **R3-1**, **R3-2** and **R3-3**, renaming them as "**R4-1**", "**R4-2**" and "**R4-3**", respectively. But we modify the rule **R3-4** and obtain a new rule "**R4-4**" described in the following.

We make the decryption oracle reject all ciphertexts $c = (c_1, c_2)$ such that the corresponding value $(m||\kappa)$ has not been queried to the random oracle H by $\mathsf{A}^{\mathsf{CCA}}$. If c is a valid ciphertext and $G(\kappa)$ has been queried then the rule of this game causes a difference: If c is valid then we have $[t]_{k_1} = [c_2 \oplus G(\kappa)]_{k_1} = H(m||\kappa)$. But we have assumed that $H(m||\kappa)$ had not been queried and hence the event $[c_2 \oplus G(\kappa)]_{k_1} = H(m||\kappa)$ happens with probability $1/2^{k_1}$ since the output of the random oracle H is uniformly distributed in $\{0,1\}^{k_1}$.

Summing up all the decryption queries, we have

$$|\Pr[\mathsf{AskKey}_4] - \Pr[\mathsf{AskKey}_3]| \leq \frac{q_D}{2^{k_1}}.$$

- Game $\mathbf{G_5}$: We retain all the rules **R4-1**, **R4-2** and **R4-3**, renaming them as "**R5-1**", "**R5-2**" and "**R5-3**", respectively. But we put the following additional rule to **R4-4** and obtain a new rule "**R5-4**".

  We make the decryption oracle reject all ciphertexts $c = (c_1, c_2)$ such that the corresponding value κ has not been queried to the random oracle G by $\mathsf{A}^{\mathsf{CCA}}$. If c is a valid ciphertext and $H(m||\kappa)$ has been queried then the rule of this game causes a difference: If c is valid, we have $[t]_{k_1} = [c_2 \oplus G(\kappa)]_{k_1} = H(m||\kappa)$. But we have assumed that $G(\kappa)$ had not been queried and hence the event $[c_2 \oplus G(\kappa)]_{k_1} = H(m||\kappa)$ happens with probability $1/2^{k_1}$ assuming that $\mathsf{A}^{\mathsf{CCA}}$ correctly guesses the last k_1-bits of the output of the random oracle G.

  Summing up all decryption queries, we have

$$|\Pr[\mathsf{AskKey}_5] - \Pr[\mathsf{AskKey}_4]| \leq \frac{q_D}{2^{k_1}}.$$

- Game $\mathbf{G_6}$: Note that the cases when $H(m||\kappa)$ or $G(\kappa)$ has not been queried are excluded in this game since these cases were already dealt with in the previous game. That is, we assume that $H(m||\kappa)$ and $G(\kappa)$ have been queried at some point.

  We retain all the rules **R5-1**, **R5-2** and **R5-3**, renaming them as "**R6-1**", "**R6-2**" and "**R6-3**", respectively. But we add the following rule to **R5-4** and obtain a new rule "**R6-4**".

  We replace the decryption oracle by a decryption oracle simulator which can decrypt a submitted decryption query $c = (c_1, c_2)$ without knowing the private key.

Before presenting the simulator, we define some conventions. We denote by $\mathsf{GList1}$ a list which consists of simple "query-answer" entries for the random oracle G of the form $\langle \kappa, \mu \rangle$ where $\mu = G(\kappa)$. We also denote by $\mathsf{GList2}$ a list which consists of the special "query-answer" entries for the random oracle G which are of the form $c||(?, \mu)$. The symbol μ implicitly represents the query-answer relation $\mu = \mathsf{G}(\kappa)$, although the input κ is not explicitly stored and hence is denoted by "?". Note that new entries in the list $\mathsf{GList2}$ are added by the decryption oracle simulator. Similarly, we denote a list of all "query-answer" pairs for the random oracle H by HList. More specifically, HList consists of the pairs $\langle (m||\kappa), \sigma) \rangle$, where $\sigma = H(m||\kappa)$. Notice that all these lists are growing as $\mathsf{A}^{\mathsf{CCA}}$'s attack proceeds.

Now, we describe a complete specification of the decryption oracle simulator. Note in the following that the decryption oracle simulator can be constructed using $\mathsf{A}^{\mathsf{GDH}}$'s DDH oracle $\mathcal{O}_g^{DDH}(\cdot, \cdot, \cdot)$ to check whether (c_1, y, κ) is Diffie-Hellman tuple, where g is a generator of group $\mathcal{G}$, u is from the submitted ciphertext $c = (c_1, c_2)$ and y is the public key element replaced by g^b in Game $\mathsf{G_3}$.

- If there exists $\langle \kappa, \mu \rangle \in \mathsf{GList1}$ such that $\mathcal{O}_g^{DDH}(c_1, y, \kappa) = 1$
  - Compute $t = v \oplus \mu$.
  - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$ then output m, otherwise, reject c.
- Else if there exists $c||(?, \mu) \in \mathsf{GList2}$
  - Compute $t = c_2 \oplus \mu$.
  - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$ then output m, and reject c otherwise.
- Else generate μ uniformly at random from $\{0,1\}^{k_0+k_1}$
  - Compute $t = c_2 \oplus \mu$.
  - Put $c||(?, \mu)$ into $\mathsf{GList2}$.
  - If there exists $\langle (m||\kappa), \sigma \rangle \in \mathsf{HList}$ such that $\mathcal{O}_g^{DDH}(c_1, y, \kappa) = 1$ and $m = [t]^{k_0}$ and $\sigma = [t]_{k_1}$ then output m, and reject c otherwise.

Note that the above decryption oracle simulator perfectly simulates the real decryption oracle since $H(m||\kappa)$ and $G(\kappa)$ have been previously queried before the current game starts. Thus, we get

$$\Pr[\mathsf{AskKey_6}] = \Pr[\mathsf{AskKey_5}].$$

As defined in Game $\mathbf{G_3}$, $\mathsf{AskKey_6}$ denotes the event that the Diffie-Hellman key $y^a(= g^{ab})$ has been queried to the random oracle G or H. At this stage, we can check which one of the queries to the random oracles G and H is a Diffie-Hellman key of g^{ab} using $\mathsf{A}^{\mathsf{GDH}}$'s DDH oracle $\mathcal{O}_g^{DDH}$. Also, note that we have used the DDH oracle to simulate the decryption oracle. That is, we can solve the GDH problem and hence we have

$$\Pr[\mathsf{AskKey_6}] \leq \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{GDH}}}(k).$$

Now, putting all the bounds we have obtained in each game together, we have

$$\begin{aligned}\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{MZS}}(\mathsf{A}^{\mathsf{CCA}}) &= |\Pr[S_0] - \Pr[S_2]| \leq \frac{q_D}{2^{k_1}} + \frac{q_D}{2^{k_1}} + \Pr[\mathsf{AskKey_6}] \\ &\leq \frac{q_D}{2^{k_1-1}} + \mathbf{Succ}^{\mathrm{GDH}}_{\mathcal{G}}(\mathsf{A}^{\mathsf{GDH}}).\end{aligned}$$

Note that since $k_1 = \lambda k \in \mathbb{N}$ for some real constant $\lambda \geq 1$ as defined in the description of $\mathtt{MSZ}$, the term $\frac{q_D}{2^{k_1-1}}$ is negligible in k.

Finally, we work out the number of the calls to the DDH oracle $\mathcal{O}_g^{DDH}$ and $\mathsf{A}^{\mathsf{CCA}}$'s running time. In worst case, all of the queries to the random oracles G and H should be checked using the oracle $\mathcal{O}_g^{DDH}$ to find the correct Diffie-Hellman key of g^a and g^b. The number of these queries are bounded by $q_G + q_H$. Also, up to $q_G + q_H$ queries are made to the oracle $\mathcal{O}_g^{DDH}$ per each decryption query in the decryption oracle simulator. Therefore, the total number of calls to the DDH oracle $\mathcal{O}_g^{DDH}$ is bounded by

$$q_{DDH} \leq q_G + q_H + q_D(q_G + q_H).$$

The total running time of $\mathsf{A}^{\mathsf{GDH}}$ is bounded by

$$t' = t + q_G + q_H + q_D(q_G + q_H)(T_{DDH} + O(1)),$$

where t and T_{DDH} denote the running time of $\mathsf{A}^{\mathsf{CCA}}$ and the DDH oracle respectively. □

3.5 Implementation Issues

One important aspect of implementing the MZS scheme is how to choose a suitable group $\mathcal{G}$. The choice is quite flexible. A prime-order subgroup of the multiplicative group $\mathbb{Z}_p^*$ where p is prime would be a possible one. Also, one can choose a group of points on suitable elliptic curves instead. As is widely known, properly chosen elliptic curves can provide a highly secure public key cryptosystem with relatively small block size.

The other important issue is the implementation of the hash functions G and H, which are assumed to be random oracles [20]. Recall that the inputs of the hash function H are of the form $(m||\kappa)$, where m is a bit string of length k_0 and κ is an element of the group $\mathcal{G}$, represented as an integer. Since applications usually work with data represented as octet (byte) strings rather than bit strings, we need to convert any bit strings or integers used as variables in the scheme into octet strings. According to the IEEE P1363 standard [67], we can use the *BS2OSP* function to convert a bit string to an octet string. Also, using the *I2OSP* function, we can convert an integer to an octet string. Therefore, $(m||\kappa)$ in the scheme is actually implemented as $(BS2OSP(m)||I2OSP(\kappa))$.

Construction of the hash function H as a random oracle can be quite flexible but we recommend the Key Derivation Function 1 (KDF1) defined in the IEEE P1363 standard.

Using the KDF1, H can be implemented as follows.

$$H(M) = hash(M||I2OSP(0))|| \cdots ||hash(M||I2OSP(l-1)),$$

where $hash$ is a conventional hash function such as SHA-1 [87] with output length $hLen$, $M \stackrel{\text{def}}{=} (BS2OSP(m)||\ I2OSP(\kappa))$ and $l = \lceil k_1/hLen \rceil$.

The hash function G can be implemented in a similar manner except for using a different conventional hash algorithm for $hash$, such as MD5 [102].

3.6 Discussions on SQS's Security Analysis

At ACISP 2002, Soldera, Qu, and Seberry (SQS) [115] proposed a slightly different modification of the Zheng-Seberry scheme called "Secure ElGamal (SEG)", which can be described as follows.

Let $pk = (\mathcal{G}, g, q, y)$, where $y = g^x$, and $sk = (\mathcal{G}, g, q, x)$. Now, a plaintext message m is encrypted by creating a ciphertext

$$c = (u, v) = (g^r, y^r \cdot (m||H(m||y^r))^2),$$

where r is randomly chosen from $\mathbb{Z}_q^*$ and H is a hash function modelled as a random oracle. Decryption of c is similar to the `MZS` scheme, so we omit it here.

We now look into the proof of the SEG scheme. According to [115], the SEG scheme described above is provably secure in the sense of IND-CCA in the random oracle model assuming the DDH problem is intractable. However, we show that the proof is incorrect.

Recall that in the DDH problem, given a tuple (g^a, g^b, g^c) and a generator g, the attacker is to decide whether $c = ab$. In the security proof of the SEG scheme, a variant of the DDH is used. In this variation, the attacker is to distinguish the distribution of random quadruples $D = (g_1, g_2, u_1, u_2) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}$ from the distribution of quadruples $R = (g_1, g_2, u_1, u_2) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G} \times \mathcal{G}$, where g_1 and g_2 are random, and $u_1 = g_1^r$ and $u_2 = g_2^r$ for random $r \in \mathbb{Z}_q^*$. If we let $g_1 = g$ and $g_2 = g^s$ then the distribution D becomes (g, g^s, g^r, g^{sr}) and hence the above "distinguishing D from R" problem becomes the standard DDH problem.

To show that the SEG scheme is secure in the sense of IND-CCA assuming that the DDH problem is hard, we should simulate the view of the IND-CCA attacker up to the point where we get the ability of the attacker solving the DDH problem to achieve its goal.

In [115], the encryption oracle of the SEG scheme is simulated as follows: We choose a private key x_R randomly from $\mathbb{Z}_q^*$ and compute $y_{sim} = g^{x_R}$. Then, we give $(\mathcal{G}, g, q, y_{sim})$ as a public key to the IND-CCA attacker. On receiving a plaintext pair (m_0, m_1) from the IND-CCA attacker, we select $\beta \in \{0, 1\}$ at random and output a target ciphertext (c_1, c_2, c_3), where $c_1 = u_1$, $c_2 = u_2$ and $c_3 = c_1^{x_R} \cdot c_2 \cdot (m_\beta||H(m_\beta||c_1^{x_R}))^2$. Notice that u_1 and u_2 are from the distribution D or R.

Obviously, there is a serious problem in the above simulation of the encryption oracle. Note in the above simulation that the simulated target ciphertext has an extra component c_2. Since c_2 is an element of the group $\mathcal{G}$, the length of the simulated target ciphertext (c_1, c_2, c_3) is always 1.5 times longer than that of the target ciphertext in the real attack. Hence, with non-negligible probability, the IND-CCA attacker can distinguish the target ciphertext in the simulation from that in the real attack by telling which one is longer.

To make the above simulation correct, we should convert (c_1, c_2, c_3) to one that has the same length of the target ciphertext in the real attack. But, how to do this is not precisely mentioned anywhere in the proof except the very vague statement that "the transformation back is obvious": Actually, throughout the entire proof, they use the lengthened target ciphertext (c_1, c_2, c_3) as if it were a correctly simulated one.

There is another problem. It is claimed in the proof that "$c_1^{x_R} c_2$ is equivalent to the output of the actual encryption oracle". However, we show in the following that this is not the case.

In the real attack, the target ciphertext is of the form

$$(u, v) = (g^r, y^r \cdot (m||H(m||y^r))^2),$$

where $y = g^x$, hence, the component y^r yields a correct Diffie-Hellman key of $u = g^r$ and $y = g^x$ since $y^r = g^{xr}$. On the other hand, in the simulation, even if u_1 and u_2 are from the distribution D, $c_1^{x_R} c_2$ does not yield a Diffie-Hellman key of any combination of y_{sim}, c_1 and c_2: Suppose that $g_1 = g$ and $g_2 = g^s$ for random $s \in \mathbb{Z}_q^*$. Since $u_1 = g_1^r$ and $u_2 = g_2^r$, we have $u_1 = g^r$ and $u_2 = g^{sr}$. Now, we obtain

$$c_1^{x_R} c_2 = u_1^{x_R} u_2 = g^{r x_R} g^{sr} = g^{r x_R + sr}.$$

Thus, the distribution $(y_{sim}, c_1, c_2, c_1^{x_R} c_2)$ is equivalent to $(g^{x_R}, g^r, g^{sr}, g^{r x_R + sr})$. However, any three components of this distribution do not yield a Diffie-Hellman tuple.

The implication of the above problem is even more serious: Since the combination $(g^r, g^{sr}, g^{r x_R + sr})$ is not a Diffie-Hellman tuple, we cannot hope to solve the problem of distinguishing D from R, that is, the DDH problem, using the ability of the IND-CCA attacker.

So far, we have discussed the problem of SQS's security analysis of the SEG given in [115] rather informally. However, motivated by the work of Steinfeld, Baek, and Zheng [117], it can be formally shown that the intractability of the "Strong Diffie-Hellman (SDH)" problem, which is a variant of the GDH problem, is a *necessary* condition for the scheme SEG to be IND-CCA secure in the random oracle model, which in effect supports our claim given in the previous section: If the proof in [115] were true (that is, the SEG scheme were IND-CCA secure assuming that the DDH problem is computationally hard), it would be the case that there exists a reduction from the DDH problem to the SDH problem. But

this is impossible in that we are unable to simulate the DDH oracle that the attacker for the SDH problem has access to [117].

As mentioned above, the SDH problem is a variant of the GDH problem defined in Section 3.2.2. In the SDH problem, the attacker has access to the special DDH oracle which on receiving (g^u, g^w), tells whether $w = ub$ for a *fixed* group element g^b. Given (g, g^a, g^b), the attacker's goal is, of course, to compute g^{ab}. Below, we define the SDH problem more formally.

Definition 5 (SDH) Let $\mathcal{G}$ be a cyclic group of order $q \geq 2^k$ generated by $g \in \mathcal{G}$, where q is a prime and k is a security parameter. Let $\mathsf{A}^{\mathsf{SDH}}$ denote an attacker assumed to be a probabilistic Turing machine taking the security parameter k as input. Suppose that a and b are uniformly chosen at random from $\mathbb{Z}_q^*$ and g^a and g^b are computed.

$\mathsf{A}^{\mathsf{SDH}}$ is to solve the following problem:

- Given $(\mathcal{G}, q, g, g^a, g^b)$, compute the Diffie-Hellman key g^{ab} of g^a and g^b with the help of the fixed Decisional Diffie-Hellman (fDDH) oracle $\mathcal{O}^{fDDH}_{g,g^b}(\cdot,\cdot)$, which, given (g^u, g^w) where g is the (fixed) generator of $\mathcal{G}$, outputs 1 if $w = ub$ and 0 otherwise.

We define $\mathsf{A}^{\mathsf{SDH}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{SDH}}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{SDH}}(\mathcal{G}, q, g, g^a, g^b) = g^{ab}].$$

We denote by $\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G}}(t_{SDH}, q_{fDDH})$ the maximal success probability $\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{SDH}}}(k)$ over all attackers $\mathsf{A}^{\mathsf{SDH}}$ having running time t_{SDH} and the number of queries to the fDDH oracle is bounded by q_{fDDH}. The running time t_{fDDH} and the number of fDDH oracle queries q_{fDDH} are polynomial in the security parameter k.

The SDH problem is said to be intractable in the group $\mathcal{G}$ if $\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G}}(t_{SDH}, q_{fDDH})$ is negligible in k.

We now state and prove the following theorem.

Theorem 2 *The intractability of the SDH problem is a necessary condition for the scheme SEG to be IND-CCA secure in the random oracle model. More precisely, we obtain the following bound:*

$$\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G}}(t_{SDH}, q_{fDDH}) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{SEG}}(t_{CCA}, q_H, q_D) + \frac{q_{fDDH} + 1}{2^{k_1}},$$

where $t_{CCA} = t_{SDH} + q_{fDDH}O(1) + O(k^3)$, $q_H = q_{fDDH} + 1$, *and* $q_D = q_{fDDH}$ *for the security parameter* $k \in \mathbb{N}$.

Note that q_{fDDH} *denotes the number of queries made by the SDH attacker having running time* t_{SDH} *to the fDDH oracle* $\mathcal{O}^{fDDH}_{g,g^b}$. *Note also that* q_H *and* q_D *denote the number of queries made by the IND-CCA attacker having running time* t_{CCA} *to the random oracle* H *and the decryption oracle respectively.*

Proof. Let $\mathsf{A}^{\mathsf{SDH}}$ denote an attacker for solving the Strong Diffie-Hellman problem (Definition 5). Let $\mathsf{A}^{\mathsf{CCA}}$ denote an attacker that defeats the IND-CCA security of the SEG scheme. Assume that $\mathsf{A}^{\mathsf{SDH}}$ is given $(\mathcal{G}, q, g, g^a, g^b)$.

We now define a game $\mathbf{G_0}$ which is exactly the same as the game for SDH problem described in Definition 5.

- Game $\mathbf{G_0}$: We denote by S_0 the event that $\mathsf{A}^{\mathsf{SDH}}$ outputs the Diffie-Hellman key g^{ab} of g^a and g^b. Denote the same event in Game $\mathbf{G_i}$ by S_i.

  Sine this game is the same as the real attack game of the SDH problem, we obtain

  $$\Pr[S_0] = \mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{SDH}}}(k).$$

- Game $\mathbf{G_1}$: In this game, we first replace g^b by $\mathsf{A}^{\mathsf{CCA}}$'s public key $y = g^x$ of the SEG scheme. We then get $\mathsf{A}^{\mathsf{CCA}}$'s target ciphertext $c^* = (c_1^*, c_2^*) = (g^{r^*}, y^{r^*} \cdot (m_\beta || H(m_\beta || y^{r^*}))^2)$, where β is a random bit and replace g^a by c_1^*. Since we have just replaced the set of random variables by another set of random variables which is identically distributed, we obtain

  $$\Pr[S_1] = \Pr[S_0].$$

- Game $\mathbf{G_2}$: In this game, we replace $\mathsf{A}^{\mathsf{SDH}}$'s fDDH oracle $\mathcal{O}^{fDDH}_{g,g^b}$ by the following simulator which uses the decryption oracle of the SEG scheme that $\mathsf{A}^{\mathsf{CCA}}$ has access to.

  - For each of new query (g^u, g^w) where
    - Set $c_1 = g^u$.
    - Choose $m \neq m_\beta$ arbitrarily from $\{0,1\}^{k_0}$.
    - Compute $\sigma = H(m||g^w)$.
    - Compute $c_2 = \kappa(m||\sigma)^2$.

- Query (c_1, c_2) as a ciphertext to the decryption oracle of the SEG scheme, which decrypts the ciphertext under the private key x.
- If the decryption oracle returns a "*Reject*" message then output 0, otherwise output 1.

Now, let $\mathsf{DDHSimErr}$ be the event that $g^w \neq c_1^x$, where x is a private key corresponding to y, but the simulator described above outputs 1.

Unless $\mathsf{DDHSimErr}$ happens, there is no difference between game $\mathsf{G_2}$ and $\mathsf{G_1}$. Thus, we have $|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{DDHSimErr}]$.

It remains to upper bound $\mathsf{DDHSimErr}$. Note that if the event $\mathsf{DDHSimErr}$ occurs, it is the case that $\sigma(= H(m||g^w)) = H([t]^{k_0}||c_1^x)$ where $t = \sqrt{c_2/c_1^x}$ and $g^w \neq c_1^x$, which happens with probability $1/2^{k_1}$. Since $\mathsf{A}^{\mathsf{GDH}}$ makes queries up to q_{fDDH}, we have $\Pr[\mathsf{DDHSimErr}] \leq \frac{q_{fDDH}-1}{2^{k_1}}$.

$$|\Pr[S_2] - \Pr[S_1]| \leq \frac{q_{fDDH}}{2^{k_1}}.$$

- Game $\mathbf{G_3}$: The rules of this game are the same as Game $\mathbf{G_2}$ except when $\mathsf{A}^{\mathsf{SDH}}$ outputs κ^*. If that event happens, we perform the following:
  - Compute $t = \sqrt{c_1^*/\kappa^*}$. (Let $[t]^{k_0}$ be the first k_0-bits of t, starting with the first bit. Let $[t]_{k_1}$ be the remaining k_1-bits of t, starting with the (k_0+1)-th bit.)
  - Compute $\sigma^* = H([t]^{k_0}||\kappa^*)$.
  - If $[t]_{k_1} = \sigma^*$, output $\beta' \in \{0,1\}$ such that $[t]^{k_0} = m_\beta$ and $\beta' = \beta$.
  - Otherwise, choose β' at random from $\{0,1\}$ and output it.

  As in the previous game, we consider the event that $\sigma^* = H([t]^{k_0}||\kappa^*) = H([t']^{k_0}||\kappa')$, where $\kappa^* \neq \kappa'$, which happens with probability $1/2^{k_1}$. This events makes differences between this game and the previous one, hence we have

$$|\Pr[S_3] - \Pr[S_2]| \leq \frac{1}{2^{k_1}}.$$

Now, splitting the even "$\beta' = \beta$", we get

$$\Pr[\beta' = \beta] = \Pr[\beta' = \beta | S_3] \Pr[S_3] + \Pr[\beta' = \beta | \neg S_3] \Pr[\neg S_3].$$

Since $\Pr[\beta' = \beta | S_3] = 1$ and $\Pr[\beta' = \beta | \neg S_2] = \frac{1}{2}$, we have

$$\Pr[\beta' = \beta] = \Pr[S_3] + \frac{1}{2}\Pr[\neg S_3] = \frac{1}{2} + \frac{1}{2}\Pr[S_3].$$

Thus, we have

$$\Pr[S_3] = 2\Pr[\beta' = \beta] - 1 = \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{SEG},\mathsf{A}^{\mathsf{CCA}}}(k).$$

Putting all the bounds we have obtained in each game together, we get the following.

$$\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{SDH}}}(k) \leq \frac{q_{fDDH}}{2^{k_1}} + \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{SEG},\mathsf{A}^{\mathsf{CCA}}}(k)$$

Considering the running time t_{SDH} and the number of queries q_{fDDH} to the fDDH oracle, we obtain

$$\mathbf{Succ}^{\mathrm{SDH}}_{\mathcal{G}}(t_{SDH}, q_{fDDH}) \leq \mathbf{Succ}^{\mathrm{IND-CCA}}_{\mathtt{SEG}}(t_{CCA}, q_H, q_D) + \frac{q_{fDDH}}{2^{k_1}},$$

where $t_{CCA} = t_{SDH} + q_{fDDH}O(1) + O(k^3)$, $q_H = q_{fDDH} + 1$ and $q_D = q_{fDDH}$.

□

3.7 Brief Summary of the Results

In this chapter, we formally proved that the intractability of the *GDH* problem is sufficient for the modified Zheng-Seberry scheme `MZS` to be secure against chosen ciphertext attack in the random oracle model. Through informal and formal arguments, we also pointed out the flaws in Soldera et al.'s [115] security analysis of Zheng and Seberry's scheme.

Chapter 4

Security Analysis of Signcryption

4.1 Introduction

4.1.1 Motivation

Over the last decades, a number of public key encryption schemes have emerged, and evolved as demands for *efficient* schemes that meet a *strong confidentiality requirement* such as chosen ciphertext security grow. The notable contributions in this line of research include Zheng and Seberry's [129] practical approaches to construct public key encryption schemes that resist chosen ciphertext attack; Bellare and Rogaway's [18] padding method for giving the RSA function chosen ciphertext security, which is known as "OAEP"; Fujisaki and Okamoto's [55] transformation that converts any IND-CPA secure public key encryption schemes to IND-CCA secure ones; and Cramer and Shoup's construction of a public key encryption scheme provably secure against chosen ciphertext attack without relying on the random oracle model [20].

Similarly to public key encryption schemes, constructing efficient digital signature schemes that meet a *strong authenticity requirement* such as unforgeability against chosen message attack, has also been regarded as of prime importance. Early constructions of the efficient digital signature schemes include "RSA" [103], "ElGamal" [49], and "Schnorr" [104]. These schemes with some modifications were later proved to be secure in the random oracle by Bellare and Rogaway [19] (the RSA signature scheme from [103]), and Pointcheval and Stern [97] (the ElGamal and Schnorr signature schemes from [49] and [104] respectively) under the notion of existential unforgeablity against chosen message attack formalized by Goldwasser, Micali, and Rivest [63].

A natural question one can now ask is how to integrate public key encryption and digital signature in an efficient way without sacrificing each other's security, in other words, how to efficiently provide communicating messages with confidentiality and authenticity *simultaneously* as one secure cryptographic function. In 1997, Zheng [124] gave a positive answer to the above question: He proposed a significantly *efficient* cryptography scheme called *signcryption* which combines the functionality of discrete-logarithm based public key encryption and digital signature schemes.

Although Zheng's original signcryption scheme has been focused by a number of research works, e.g. [12, 58, 79, 106, 65, 126, 127], no security analysis of Zheng's signcryption through a framework of the *provable security* approach, as far as we know, has been. In this chapter, we present precise confidentiality and unforgeablity models for generic signcryption schemes and provide rigorous proofs, based on these models, that Zheng's original signcryption scheme meets strong security requirements with respect to message confidentiality and unforgeability under known cryptographic assumptions.

4.1.2 Related Work

Compared with the public key setting, research on the integration of message confidentiality and authenticity has been quite active in the symmetric setting. A series of research works has appeared on using modes of block ciphers to give both message confidentiality and integrity [70, 101]. Also, security issues related to the composition of symmetric key encryption and message authentication code (MAC) were considered by Bellare and Namprepre [17]. They concluded that only "Encrypt-then-MAC (EtM)" composition is *generically* secure against chosen ciphertext attack and existentially unforgeable against chosen message attack. Krawczyk [74] also considered the same problem when building a secure channel over insecure networks. Interestingly, his conclusion was that the "MAC-then-Encrypt (MtE)" composition is also secure, under the assumption that encryption method is either a secure CBC mode or a stream cipher that XORs the data with a random pad.

In the public key setting, Tsiounis and Yung [121] proposed a variant of the ElGamal encryption scheme where Schnorr's signature is used to provide non-malleability. However, the security goal of their scheme is to provide confidentiality, consequently the strong origin authentication is not supported in their scheme. We remark that this scheme was recently analyzed again by Schnorr and Jakobsson [105] under the generic model plus the random oracle model. We also

remark that the security proof of Tsiounis and Yung's scheme given in [121] was later found to be flawed [111]: The Schnorr signature scheme that was used as a "proof of knowledge" in their public key encryption scheme, makes it impossible to efficiently simulate the responses to the chosen ciphertext attacker's decryption queries. (Readers are referred to [112] for more details.)

The first attempt to provide a formal security analysis of signcryption schemes was made by Steinfeld and Zheng [116]. They proposed a signcryption scheme based on the integer factorization problem and provided a formal security model and security proof for the unforgeability of their scheme. But, the same for the confidentiality was not provided. (We remark, however, that following the work presented in this chapter, the analysis of the factoring-based signcryption scheme in [116] has been extended to cover both confidentiality and unforgeability in the "Multi-User" setting [2] that will be discussed later in this chapter.)

Recently and independently of our work, similar security notions to ours were defined by An, Dodis and Rabin (ADR) [2], who analyzed the security of *generic* compositions of black-box signature and public key encryption schemes. Our unforgeability notion FiSO-UF-CMA, which will be presented precisely in Section 4.2.3, corresponds to the "Two-User Insider" setting defined by ADR in [2], whereas our confidentiality notion FSO/FUO-IND-CCA, which will be defined precisely in Section 4.2.2, corresponds to the weaker "Multi-User Outsider" setting of ADR. In Section 4.1.3, we discuss our models and their relationship to those defined by ADR in greater detail.

4.1.3 Differences between Our Security Model and Other Models

Differences between Symmetric and Public Key Models

To address the significant difference between security implication of the compositions of encryption and authentication in the symmetric setting and that in the public key setting, we consider confidentiality of the "Encrypt-then-MAC (EtM)" and "Encrypt-and-MAC (EaM)" compositions in the symmetric setting, and the security of the directly corresponding simple public key versions "Encrypt-then-Sign (EtS)" and "Encrypt-and-Sign (EaS)" (defined in the natural way, with the signer's public key appended). We point out that while the symmetric composition EtM is secure against chosen ciphertext attack (indeed, EtM is generically secure as shown in [17]), the simple public key version EtS is *completely insecure against chosen ciphertext attack*, even if the underlying encryption scheme

is secure against chosen ciphertext attack. The reason is that in the public key versions, a ciphertext in the composed scheme contains an additional component (which does not present in the symmetric versions), namely the *sender's signature public key*. The fact that this component is easily malleable implies the insecurity of the public key version EtS under chosen ciphertext attack.

As an example, assume that Alice encrypts and signs her message m following the EtS composition. That is, she encrypts the message m using a public key encryption algorithm $\mathsf{E}_{pk_B}(\cdot)$ and computes $c = \mathsf{E}_{pk_B}(m)$. Then she signs on c using her digital signature algorithm $\mathsf{S}_{sk_A}(\cdot)$ to produce $\sigma = \mathsf{S}_{sk_A}(c)$. Now the ciphertext C is (c, σ). However, an attacker Marvin now generates his own public and private key pair (pk_M, sk_M) and signs on c obtained by eavesdropping the ciphertext C en route from Alice to Bob. Namely, she can produce $C' = (c, \mathsf{S}_{sk_M}(c))$ where S_{sk_M} is Marvin's digital signature algorithm. Then he hands in his public key pk_M (which may be contained in Marvin's digital certificate) to Bob. Now notice that C' which is different from C is completely verified as a valid ciphertext using Marvin's public key pk_M and Bob decrypts it into m. Hence Marvin succeeds in his chosen ciphertext attack on the EtS scheme even if the underlying public key encryption scheme is strong, say, secure against chosen ciphertext attack.

Discussion of Our Models in the Context of Other Public Key Models

We now discuss the relationship between security models for signcryption schemes defined by An, Dodis and Rabin (ADR) [2] and our security notions as defined in Section 4.2. First, we review the classification of security models for signcryption schemes defined by ADR [2].

Two-User and Multi-User Settings. The first classification of security models for signcryption schemes depends on the assumed application setting. In the *Two-User* setting, it is assumed that there are only two users of the scheme: a single *sender* Alice with key pair (sk_A, pk_A) and a single *receiver* Bob with key pair (sk_B, pk_B). Hence in this setting, the receiver's public key that Alice uses to signcrypt all her messages is Bob's public key pk_B which is *fixed* for the entire session. Similarly, the sender's public key that Bob uses to unsigncrypt all his signcryptexts Alice's public key pk_A which is also fixed for the session. In contrast, in the *Multi-User* setting, we assume that there are *many users* of the scheme besides the attacked users Alice and Bob. Therefore in this setting, the receiver's public key that Alice uses to signcrypt her messages can be *any* receiving party's public key pk_R (not necessarily Bob's pk_B). Similarly, the sender's public key

that Bob uses to unsigncrypt his signcryptexts can be any sending party's public key pk_S (not necessarily Alice's pk_A). In particular, in this setting the attacker is given the power to choose his own receiver and sender's public keys when accessing Alice and Bob's signcryption and unsigncryption oracles respectively. This power does not exist in the Two-User setting.

Insider and Outsider Settings. The second classification of security models for signcryption schemes depends on the identity of the attacker. In the *Outsider* setting, the attacker is assumed to be a *third-party* distinct from both the attacked users Alice and Bob. In order to break confidentiality in this setting, the goal of the attacker is to recover some information on a message signcrypted by Alice to Bob, assuming the signcryptext has not been unsigncrypted by Bob. To break unforgeability in this setting, the goal of the attacker is to forge a signcryptext from Alice to Bob on a message which has not been signcrypted by Alice. Note that in the outsider setting, since the attacker is a third-party, he only knows the *public* keys of Alice and Bob. In contrast, in the *Insider* setting, the attacker is assumed to be a *second-party*, meaning that the attacker is either Alice (attacking Bob's confidentiality) or Bob (attacking Alice's unforgeability). To break Bob's confidentiality in this setting, Alice's goal is to obtain some information on a message signcrypted to Bob with Alice's public key as the sender's public key, assuming the signcryptext has not been unsigncrypted by Bob with Alice's public key as the sender's public key (note that in this setting, the attacker Alice may know the sender's secret key). To break Alice's unforgeability in this setting, Bob's goal is to forge a valid signcryptext from Alice to Bob on a message which has never been signcrypted by Alice to Bob (note that in this setting, the attacker Bob may know the receiver's private key).

Relation to Our Confidentiality Notion. One can easily notice that the strongest confidentiality notion for signcryption schemes is the confidentiality in the "Multi-User Insider" setting. However, Zheng's original signcryption scheme [124] is completely insecure in this setting because the Diffie-Hellman key $g^{x_A x_B}$ (which is easily recoverable by the sender Alice) from Alice and Bob's public keys g^{x_A} and g^{x_B} suffices to unsigncrypt *any* signcryptexts from Alice to Bob. As also discussed in [2], this model, however, is not of significant importance in normal circumstances since it in effect assumes that the sender Alice is trying to decrypt a signcryptext which was sent by herself. Hence, this model appears only useful in special situations where an attacker who breaks into Alice's system obtains her private key in order to decrypt a message previously signcrypted by Alice to Bob. But, this insecurity can be considered as a positive feature, called *past message recovery* by Zheng [125], whereby Alice can store past signcryptexts and unsigncrypt them in the future when desired.

From the discussions above, we believe that for most applications it suffices for a signcryption scheme to achieve confidentiality in the "Multi-User Outsider" setting. Our independently defined confidentiality notion "FSO/FUO-IND-CCA" for this setting matches the corresponding definition by ADR [2].

Relation to Our Unforgeability Notion. Since signcryption offers unforgeablity of signcryptext for the sender Alice, it is essential that even the receiver Bob cannot impersonate Alice and forge valid signcryptexts from Alice to himself. To ensure that our unforgeability notion for signcryption covers this aspect, we allow the forger in our attack model to have access to Bob's private key as well as the corresponding public key. That is, our unforgeablity notion corresponds to the unforgeability in the "Insider" setting. In this thesis, we only consider the case when two users are involved in the signcryption process, so our independently defined unforgeability notion, which we call "FiSO-UF-CMA", matches the unforgeablity in the "Two-User Insider" setting of ADR. (We remark, however, that in our recent paper [7], the FiSO-UF-CMA notion has been extended to the even stronger notion that matches the unforgeability in the "Multi-User Insider" setting, and has been shown that Zheng's original signcryption scheme is still secure under this notion.)

Like the model proposed by ADR [2], our model also does not explicitly provide *non-repudiation*, meaning the ability of a receiver of a valid signcryptext to convince a third-party that a given sender has sent this signcryptext. However, as also pointed out in [2], unforgeability in the FiSO-UF-CMA sense guarantees that the receiver cannot forge any valid signcryptext by the sender, so non-repudiation can always be achieved using a protocol running between the receiver and the third-party, which convinces the third-party of the *validity* of a signcryptext with respect to a given message and sender and receiver public keys. A solution which does not compromise the receiver's private key to the third-party, is to use a zero-knowledge proof of signcryptext validity. Specific protocols for Zheng's scheme are presented by Zheng in [125].

On the Power of Attackers in the Multi-User Setting. The extra power given to the attacker in the Multi-User setting is the ability to access *flexible signcryption oracle* which allows the attacker to specify a receiver's public key in addition to a message, and the *flexible unsigncryption oracle* which allows the attacker to specify a sender's public key in addition to a signcryptext. In practice, such an attack might be mounted by the attacker Marvin by requesting a new public key certificate from the Certificate Authority (CA) each time he wants to query Alice's signcryption oracle with a new public key of his choice. Hence, signcryption

schemes that meet our confidentiality notion remain secure under such a strong attack.

4.1.4 Contributions of This Chapter

One of the most attractive features of Zheng's original signcryption scheme might be its efficiency in terms of computation. More precisely, the dominant cost in both signcryption and unsigncryption algorithms is approximately only a *single* exponentiation in the underlying subgroup. This high efficiency is achieved by *sharing* this exponentiation for both the encryption and signature "portions" of the computation, and is therefore at least 2 times more efficient than a generic composition (using one of the generic compositions presented in [2]) of discrete-logarithm based signature and encryption schemes, each of which would presumably perform (at least) one separate exponentiation.

Our results demonstrate that despite its high efficiency, Zheng's scheme still meets strong security requirements with respect to known cryptographic assumptions and the random oracle model for the underlying hash functions. Specifically, our main results can be summarized as follows. First, we prove, in the random oracle model, that Zheng's original signcryption scheme meets our confidentiality notion for signcryption schemes, "FSO/FUO-IND-CCA", which corresponds to ADR's confidentiality notion in the "Multi-User Outsider" setting, under the assumption that the Gap Diffie-Hellman (GDH) problem [90] in a prime-order subgroup of $\mathbb{Z}_p^*$ for p prime is computationally intractable, and the assumption that the underlying one-time symmetric encryption scheme is secure. Second, we prove, in the random oracle model, that Zheng's scheme meets our unforgeability notion, "FiSO-UF-CMA", which corresponds to ADR's unforgeability notion in the "Two-User Insider" setting, assuming that the Discrete-Logarithm problem in the underlying subgroup is computationally hard.

Of course, one should remark that a disadvantage of Zheng's scheme is its reliance on specific number-theoretic computational complexity assumptions, and on the random oracle model. These assumptions can both be avoided, at the cost of efficiency, by using a generic encryption-signature composition scheme and applying ADR's results in [2].

4.2 Our Security Notions for Signcryption

4.2.1 Formal Definition of Generic Signcryption

First, we formally define a generic signcryption scheme, which we denote by "$\mathcal{SCR}$".

Definition 6 (Generic Signcryption) A generic signcryption scheme $\mathcal{SCR}$ consists of the following algorithms:

- A randomized common parameter generation algorithm $\mathsf{GC}(k)$: Given a security parameter $k \in \mathbb{N}$, this algorithm generates a set of public common parameters, e.g., descriptions of hash functions and a mathematical group. The output of this algorithm denoted by cp includes such parameters as well as the security parameter k.
- A randomized user key-pair generation algorithm $\mathsf{GK}(cp)$: Given a security parameter cp, this algorithm generates a single user's key pair (sk, pk) where sk and pk denote the private key and public key respectively.
- A randomized signcryption algorithm $\mathsf{SC}(cp, sk_A, pk_B, m)$: Given a common parameters sequence cp, a sender's secret key sk_A, a receiver's public key pk_B, and a message $m \in SP_m$ (SP_m is the message space), this algorithm generates a signcryptext C.
- A deterministic unsigncryption algorithm $\mathsf{USC}(cp, sk_B, pk_A, C)$: Given a common parameters sequence cp, a receiver's secret key sk_B, a sender's public key pk_A, and a signcryptext C, this algorithm returns either a message m or a "*Reject*" symbol.

Figure 4.1 illustrates a schematic outline of a generic signcryption scheme.

4.2.2 FSO/FUO-IND-CCA: Our Confidentiality Notion for Signcryption

Following the discussions in Section 4.1.3, we provide an attack model for confidentiality of the generic signcryption scheme $\mathcal{SCR}$, which we call the "Flexible Signcryption Oracle/Flexible Unsigncryption Oracle (FSO/FUO)"-model. In this model, the attacker Marvin's goal is to break the confidentiality of messages

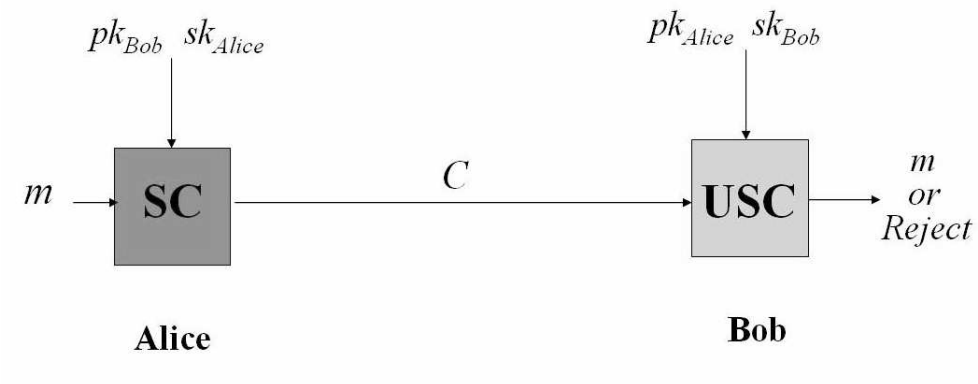


Figure 4.1: Generic Signcryption

between the sender Alice and the receiver Bob. Marvin is given Alice's public key pk_A^* and Bob's public key pk_B^*, and has access to a *flexible* signcryption oracle, as well as a *flexible* unsigncryption oracle: On receiving (pk_R, m) where pk_R denotes a receiver's public key generated by Marvin at will (Marvin may choose the receiver's public key as Bob's public key pk_B^*, say, $pk_R = pk_B^*$.) and m denotes a plaintext, the flexible signcryption oracle returns a signcryptext after performing signcryption under Alice's private key sk_A^*. On the other hand, the flexible unsigncryption oracle, on receiving (pk_S, C) where pk_S denotes a sender's public key generated by Marvin at will (Similarly to the flexible signcryption oracle, Marvin may choose the sender's public key as Alice's public key pk_A^*, say, $pk_S = pk_A^*$.) and C denotes a signcryptext, returns a plaintext after performing unsigncryption under Bob's private key sk_B^*. In other words, the flexible signcryption and unsigncryption oracles are not constrained to be executed only under pk_B^* and pk_A^* respectively – Bob and Alice's public key can be replaced by the public keys generated by Marvin. Accordingly, the FSO/FUO-model gives Marvin the full chosen ciphertext power with the ability to choose the sender and receiver's public key as well as the signcryptext.

Using the notion of indistinguishability (abbreviated by "IND") of encryptions [62], we formalize the security against chosen ciphertext attack for signcryption with respect to the FSO/FUO-model. Following the presentation style in [16], we call our notion "FSO/FUO-IND-CCA". Below, we formally define FSO/FUO-IND-CCA.

Definition 7 (FSO/FUO-IND-CCA) Let $\mathcal{SCR} = (\mathsf{GC}, \mathsf{GK}, \mathsf{SC}, \mathsf{USC})$ be a generic signcryption scheme. Let $\mathsf{A}^{\mathsf{CCA}}$ be an attacker modelled as a probabilistic

Turing machine taking the security parameter k as input. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{CCA}}$ interacts with the "Challenger".

Phase 1: The Challenger runs the common parameter generation algorithm $\mathsf{GC}(k)$ and obtains a common parameter cp. The Challenger then runs the user key-pair generation algorithm $\mathsf{GK}(cp)$ twice to generate Alice's private/public key pair (sk_A^*, pk_A^*) and Bob's private/public key pair (sk_B^*, pk_B^*). The Challenger gives cp, pk_A^*, and pk_B^* to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 2: $\mathsf{A}^{\mathsf{CCA}}$ submits a number of queries to the signcryption and unsigncryption oracles. Each of the queries to the signcryption oracle consists of a receiver's public key and a plaintext, which we denote by (pk_R, m). On the other hand, each of the queries to the unsigncryption oracle consists of a sender's public key and a signcryptext, which we denote by (pk_S, C). On receiving (pk_R, m), the Challenger runs $\mathsf{SC}(cp, sk_A^*, pk_R, m)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$. On receiving (pk_S, C), the Challenger runs $\mathsf{USC}(cp, sk_B^*, pk_S, m)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 3: $\mathsf{A}^{\mathsf{CCA}}$ chooses two equal-length plaintexts (m_0, m_1). On receiving these, the Challenger chooses $\beta \in \{0, 1\}$ at random, runs $\mathsf{SC}(cp, sk_A^*, pk_B^*, m_\beta)$, and gives $\mathsf{A}^{\mathsf{CCA}}$ the resulting target signcryptext C^*.

Phase 4: $\mathsf{A}^{\mathsf{CCA}}$ continues to submit a number of queries to the signcryption and unsigncryption oracles. As Phase 2, each of the queries to the signcryption oracle consists of a receiver's public key and a plaintext, which we denote by (pk_R, m). Each of the queries to the unsigncryption oracle consists of a sender's public key and a signcryptext, which we denote by (pk_S, C). However, a restriction here is that $(pk_S, C) \neq (pk_A^*, C^*)$. On receiving (pk_R, m), the Challenger runs $\mathsf{SC}(cp, sk_A^*, pk_R, m)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$. On receiving (pk_S, C), the Challenger runs $\mathsf{USC}(cp, sk_B^*, pk_S, m)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 5: $\mathsf{A}^{\mathsf{CCA}}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$.

We define $\mathsf{A}^{\mathsf{CCA}}$'s success by the probability

$$\mathbf{Succ}^{\mathrm{FSO/FUO-IND-CCA}}_{\mathsf{A}^{\mathsf{CCA}},\mathcal{SCR}}(k) = 2\Pr[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{FSO/FUO-IND-CCA}}_{\mathcal{SCR}}(t_{CCA}, q_{SC}, q_{USC})$ the maximum of the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CCA}}$ having running time t_{CCA} and making at most q_{SC} signcryption queries and q_{USC} unsigncryption queries. Note that

the running time and the number of queries are all polynomial in the security parameter k.

The $\mathcal{SCR}$ scheme is said to be FSO/FUO-IND-CCA secure if $\mathbf{Succ}_{\mathcal{SCR}}^{\text{FSO/FUO-IND-CCA}}(t_{CCA}, q_{SC}, q_{USC})$ is negligible in k.

4.2.3 FiSO-UF-CMA: Our Unforgeability Notion for Generic Signcryption Schemes

We now present our unforgeability notion of generic signcryption. We call this notion "FiSO-UF-CMA", which represents the unforgeability against chosen message attack for signcryption with respect to the a fixed signcryption oracle model.

More precisely, the attack model for FiSO-UF-CMA can be described as follows. The forger Marvin's goal is to forge a valid signcryptext from Alice to Bob. Marvin is given Bob's private/public key pair (sk_B^*, pk_B^*) as well as Alice's public key pk_A^*. In addition, Marvin is given access to signcryption oracle which performs signcryption under Alice's private key and Bob's public key, namely $\mathsf{SC}(cp, sk_A^*, pk_B^*, \cdot)$. Marvin can choose any message m and query the signcryption oracle to get a signcryptext by Alice on message m to Bob. At the end of the attack, Marvin is considered successful in his forgery if he produces a forgery message m^* and a forgery signcryptext C^* and such that: (1) Marvin did not query the forgery message m^* to the signcryption oracle $\mathsf{SC}(cp, sk_A^*, pk_B^*, \cdot)$, and (2) C^* is a valid signcryptext from Alice to Bob, that is, $\mathsf{USC}(cp, sk_B^*, pk_A^*, C^*)$ does not reject. A formal definition for this follows.

Definition 8 (FiSO-UF-CMA) Let $\mathcal{SCR} = (\mathsf{GC}, \mathsf{GK}, \mathsf{SC}, \mathsf{USC})$ be a generic signcryption scheme. Let $\mathsf{A}^{\mathsf{CMA}}$ be an attacker (a forger) modelled as a probabilistic Turing machine taking the security parameter k as input. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{CMA}}$ interacts with the "Challenger".

> **Phase 1**: The Challenger runs the common parameter generation algorithm $\mathsf{GC}(k)$ and obtains a common parameter cp. The Challenger then runs the user key-pair generation algorithm $\mathsf{GK}(cp)$ twice to generate Alice's private/public key pair (sk_A^*, pk_A^*) and Bob's private/public key pair (sk_B^*, pk_B^*). The Challenger gives cp, pk_A^*, and (sk_B^*, pk_B^*) to $\mathsf{A}^{\mathsf{CMA}}$.
>
> **Phase 2**: $\mathsf{A}^{\mathsf{CMA}}$ submits a number of queries to the (fixed) signcryption and unsigncryption oracles. Each of the queries to the signcryption oracle consists of a message m. On the other hand, each of the queries to the

unsigncryption oracle consists of and a signcryptext C. On receiving m, the Challenger runs $\mathsf{SC}(cp, sk_A^*, pk_B^*, m)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CMA}}$. On receiving (pk_S, C), the Challenger runs $\mathsf{USC}(cp, sk_B^*, pk_A^*, C)$ and gives the resulting output to $\mathsf{A}^{\mathsf{CMA}}$.

Phase 3: $\mathsf{A}^{\mathsf{CMA}}$ outputs a forgery (m^*, C^*) such that: such that: (1) m^* was not queried to the signcryption oracle $\mathsf{SC}(cp, sk_A^*, pk_B^*, \cdot)$, and (2) C^* is a valid signcryptext from Alice to Bob, that is, $\mathsf{USC}(cp, sk_B^*, pk_A^*, C^*)$ does not reject.

We define $\mathsf{A}^{\mathsf{CMA}}$'s success by the probability

$$\mathbf{Succ}^{\text{FiSO}-\text{UF}-\text{CMA}}_{\mathsf{A}^{\mathsf{CMA}},\mathcal{SCR}}(k) = \Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*)].$$

We denote by $\mathbf{Succ}^{\text{FiSO}-\text{UF}-\text{CMA}}_{\mathcal{SCR}}(t_{CMA}, q_{SC})$ the maximum of the attacker $\mathsf{A}^{\mathsf{CMA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CMA}}$ having running time t_{CMA} and making at most q_{SC} signcryption queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The $\mathcal{SCR}$ scheme is said to be FiSO-UF-CMA secure if $\mathbf{Succ}^{\text{FiSO}-\text{UF}-\text{CMA}}_{\mathcal{SCR}}(t_{CMA}, q_{SC})$ is negligible in k.

Note that the FiSO-UF-CMA notion is the unforgeability notion given in [6]. If one uses An et al.'s term, our unforgeability notion can be said to defined in "Two-User Insider" setting, as explained in Section 4.1.3.

Finally, we remark that the reason why we do not give the attacker access to the sender's *unsigncryption* oracle is that we implicitly assume the well-established practice that users generate independent key-pairs for sending and receiving. In this setting, it is clear that the sender's unsigncryption oracle cannot help a forger because the forger can simulate such an oracle by himself.

4.3 Zheng's Original Signcryption Scheme

In this section, we give a full description of Zheng's original signcryption scheme [124].

4.3.1 Bijective One-time Symmetric Key Encryption

As a preliminary, we define a bijective one-time symmetric key encryption scheme which is used in Zheng's original signcryption scheme. Normally, one-time symmetric key encryption schemes are used in building hybrid public key encryption schemes [42]. The one-time symmetric key encryption in Zheng's signcryption scheme plays the same role as the one used in hybrid public key encryption: A symmetric key is used only once to encrypt a single message.

Definition 9 (Bijective One-time Symmetric Key Encryption) A bijective one-time symmetric key encryption scheme, denoted by "$\mathcal{OTSE}$" consists of the following algorithms:

- A deterministic encryption algorithm $\mathsf{E}(k, \tau, m)$: Given a security parameter k, a symmetric key $\tau \in SP_\tau$, and a message $m \in SP_m$, this algorithm generates a ciphertext $c \in SP_c$. (Note that SP_m, SP_τ, and SP_c denote respectively the message, key, and ciphertext spaces whose size varies as the security parameter k. Note also that the function defined by E is one-to-one on SP_m and onto SP_c.)
- A deterministic decryption algorithm $\mathsf{D}(k, \tau, c)$: Given a security parameter k, a symmetric key $\tau \in SP_\tau$, and a ciphertext $c \in SP_c$, this algorithm returns a a message $m \in SP_m$. (Note that the function defined by D is also one-to-one on SP_c and onto SP_m.)

Note that we do not need the security against chosen plaintext attack for the one-time symmetric key encryption scheme to prove the confidentiality of Zheng's signcryption scheme. An appropriate security notion for the one-time symmetric key encryption scheme will be given in a later section. Note, however, that we will use in our proof of security the fact this scheme is *bijective*, meaning in particular the decryption function is one-to-one on the ciphertext space SP_c (and hence also encryption is deterministic).

We remark that the *one-time pad* is a good candidate for implementing the above bijective one-time symmetric encryption in that it is computationally efficient and unconditionally secure. In this case, the key size can be reduced by expanding a short key using a pseudorandom generator.

4.3.2 Description of Zheng's Original Signcryption Scheme

Now, we describe Zheng's original signcryption scheme, known as "Shorthand Digital Signature Scheme (SDSS1)" [124]. To simplify the security analysis, however, we have slightly modified it. The change is that in our description, the Diffie-Hellman Key K is directly hashed by the hash function H rather than being hashed by the other hash function G first and hashed again by H as described in [124].

Zheng's original signcryption scheme, which we denote by "$\mathtt{ZSCR}$", consists of the following algorithms:

- A common parameter generation algorithm $\mathsf{GC}(k)$
  - Choose at random primes p and q such that $|p| = k$, $q > 2^{l_q(k)}$, and $q|(p-1)$, where $|\cdot|$ denotes the number of bits in the binary representation of an input; $l_q : \mathbb{N} \rightarrow \mathbb{N}$ denotes a function determining the length of q.
  - Choose a random $g \in \mathbb{Z}_p^*$ such that $\mathrm{Ord}_p(g) = q$, where $\mathrm{Ord}_p(g)$ denotes the order of g in the multiplicative group $\mathbb{Z}_p^*$.
  - Choose a hash function $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^{l_{\mathsf{G}}(k)}$. ($l_{\mathsf{G}} : \mathbb{N} \rightarrow \mathbb{N}$ is a function determining the length of the output of G.)
  - Choose a hash function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$.
  - Choose a bijective one-time symmetric key encryption scheme $\mathtt{OTSE} = (\mathsf{E}, \mathsf{D})$ with message/key/ciphertext spaces $SP_m/\{0,1\}^{l_G}/SP_c$.
  - Return $cp = (k, p, q, g, \mathsf{G}, \mathsf{H}, \mathtt{OTSE})$
- A key-pair generation algorithm $\mathsf{GK}(cp)$
  - Choose x uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $y = g^x$
  - Set $sk = (x, y)$ and $pk = y$.
  - Return (sk, pk).
- A Signcryption algorithm $\mathsf{SC}(cp, sk_A, pk_B, m)$
  - Parse sk_A as (x_A, y_A); Parse pk_B as y_B.
  - If $y_B \notin \langle g \rangle$, then return "*Reject*".
  - Choose x uniformly at random from $\mathbb{Z}_q$.

  - Compute $K = y_B^x$ and $\tau = \mathsf{G}(K)$.
  - Compute $c = \mathsf{E}_\tau(m)$ and $r = \mathsf{H}(m, y_A, y_B, K)$.
  - If $r + x_A = 0$, then return "*Reject*". Otherwise, compute $s = x/(r + x_A)$.
  - Set $C = (c, r, s)$.
  - Return C.

- A Unsigncryption algorithm $\mathsf{USC}(cp, sk_B, pk_A, C)$
  - Parse sk_B as (x_B, y_B); Parse pk_A as y_A.
  - If $y_A \notin \langle g \rangle$, then return "*Reject*".
  - Parse C as (c, r, s).
  - If $r \notin \mathbb{Z}_q$ or $s \notin \mathbb{Z}_q^*$ or $c \notin SP_c$, then return "*Reject*". Otherwise, do the following.
    * Compute $\omega = (y_A g^r)^s$; $K = \omega^{x_B}$; $\tau = \mathsf{G}(K)$.
    * Compute $m \leftarrow \mathsf{D}_\tau(c)$.
    * If $\mathsf{H}(m, y_A, y_B, K) = r$, then return m. Otherwise, return "*Reject*".

Note that the hash functions $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^{l_G(k)}$ and $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ are modelled as random oracles [20] in the security analysis. Note also that the key length of the symmetric encryption is actually $l_\mathsf{G}(k)$.

We remark that another type of Zheng's signcryption scheme, SDSS2, can be described and analyzed in a very similar manner presented in this chapter, so we only deal with the SDSS1-type of Zheng's signcryption scheme in this thesis.

4.4 Security Analysis of Zheng's Original Signcryption Scheme

In this section, we prove the confidentiality and unforgeability of Zheng's signcryption by providing reductions from known cryptographic assumptions. Although we provide a concrete analysis of our reductions, our main goal is to demonstrate the security of signcryption against polynomial-time attackers. Hence we did not attempt to optimize the insecurity bounds for our reductions.

In what follows, we review the computational primitives that will be used in our confidentiality and unforgeability proofs for Zheng's original signcryption scheme.

4.4.1 Computational Primitives

The Generalized Gap-Diffie Hellman Problem

As already introduced in the previous section, the "Gap problem" is a computational problem proposed by Okamoto and Pointcheval [90], in which an attacker is to solve an inverting problem with the help of an oracle that solves a related decisional problem.

For our proof of confidentiality of Zheng's original signcryption in FSO/FUO-IND-CCA sense, we will need a "generalized Gap Diffie-Hellman (gGDH)" problem [90] which is a slightly generalized version of the GDH problem defined in Chapter 3 in a sense that in the gGDH problem, the attacker is given, in addition to the group element g^a and g^b for random $a, b \in \mathbb{Z}_q^*$, access to a Decisional Diffie-Hellman (DDH) oracle $\mathcal{O}^{DDH}$ that given $(\bar{g}, \bar{g}^u, \bar{g}^v, z) \in \langle g \rangle \times \langle g \rangle \times \langle g \rangle \times \langle g \rangle$ checks whether $z = \bar{g}^{uv}$ or not. That is, in the gGDH problem, the DDH oracle that the attacker has access to takes a Diffie-Hellman tuptle with respect to an *unfixed* generator $\bar{g}$ rather than the fixed generator g in the GDH problem defined in the previous chapter. (However, it is possible that $\bar{g} = g$, $\bar{g}^u = g^a$, and $\bar{g}^v = g^b$.)

A precise definition of gGDH problem is as follows.

Definition 10 (gGDH) Let p and q are primes such that $|p| = k$, $q|(p-1)$, and $q > 2^{l_q(k)}$, where $l_q : \mathbb{N} \to \mathbb{N}$ denotes a function determining the length of q. Assume that $g \in \mathbb{Z}_p^*$ satisfies $\mathrm{Ord}_p(g) = q$. Let $\mathsf{A}^{\mathsf{gGDH}}$ denote an attacker assumed to be a probabilistic Turing machine taking the security parameter k as input. Suppose that a and b are uniformly chosen at random from $\mathbb{Z}_q^*$ and g^a and g^b are computed.

$\mathsf{A}^{\mathsf{gGDH}}$ is to solve the following problem:

- Given (k, p, q, g, g^a, g^b), compute the Diffie-Hellman key g^{ab} of g^a and g^b with the help of the Decisional Diffie-Hellman oracle $\mathcal{O}^{DDH}(\cdot, \cdot, \cdot, \cdot)$, which, given $(\bar{g}, \bar{g}^u, \bar{g}^v, z)$, outputs 1 if $z = \bar{g}^{uv}$ and 0 otherwise.

We quantify $\mathsf{A}^{\mathsf{gGDH}}$'s success in solving the gGDH problem by the probability

$$\mathbf{Succ}^{\mathrm{gGDH}}_{\mathsf{A}^{\mathsf{gGDH}}, \mathbb{Z}_p^*}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{gGDH}} \text{ outputs } g^{ab}].$$

We denote by $\mathbf{Succ}^{\mathrm{gGDH}}_{\mathbb{Z}_p^*}(t_{gGDH}, q_{O^{DDH}})$ the maximum of the attacker $\mathsf{A}^{\mathsf{gGDH}}$'s success over all attackers $\mathsf{A}^{\mathsf{gGDH}}$ having running time t_{gGDH} and making at most

$q_{O^{DDH}}$ queries to the oracle $\mathcal{O}^{DDH}$. Note that the running time and the number of queries are all polynomial in the security parameter k.

The gGDH problem is said to be computationally intractable if $\mathbf{Succ}^{\mathrm{gGDH}}_{\mathbb{Z}_p^*}(t_{gGDH}, q_{O^{DDH}})$ is negligible in k.

The Discrete-Logarithm Problem

For our proof of unforgeability of Zheng's original signcryption scheme, we will need the following "Discrete-Logarithm (DL)" problem.

Definition 11 (DL) Let p and q are primes such that $|p| = k$, $q|(p-1)$, and $q > 2^{l_q(k)}$ where $l_q : \mathbb{N} \rightarrow \mathbb{N}$ denotes a function determining the length of q. Assume that $g \in \mathbb{Z}_p^*$ satisfies $\mathrm{Ord}_p(g) = q$. Let A^{DL} denote an attacker modelled as a probabilistic Turing machine taking the security parameter k as input. Suppose that y is uniformly chosen at random from $\mathbb{Z}_p^*$.

A^{DL} is to solve the following problem:

- Given (k, p, q, g, y), find $a \in \mathbb{Z}_q^*$ such that $y = g^a$.

We quantify A^{DL}'s success in solving the DL problem by the probability

$$\mathbf{Succ}^{\mathrm{DL}}_{\mathsf{A}^{\mathsf{DL}},\mathbb{Z}_p^*}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{DL}} \text{ outputs } a \in \mathbb{Z}_q^*].$$

We denote by $\mathbf{Succ}^{\mathrm{DL}}_{\mathbb{Z}_p^*}(t_{DL})$ the maximum of the attacker A^{DL}'s success over all attackers A^{DL} having running time t_{DL}. Note that the running time and the number of queries are all polynomial in the security parameter k.

The DL problem is said to be computationally intractable if $\mathbf{Succ}^{\mathrm{DL}}_{\mathbb{Z}_p^*}(t_{DL})$ is negligible in k.

4.4.2 Security Notion for Bijective One-time Symmetric Encryption

We now define a security notion for the one-time symmetric key encryption scheme $\mathcal{OTSE}$ presented in Section 4.3.1. As mentioned earlier, we do not need the security against chosen-plaintext attacks for $\mathcal{OTSE}$. We merely need the security against a passive attack which we call "indistinguishability under passive attack (IND-PSA)". We now formally define IND-PSA as follows.

Definition 12 (IND-PSA) Let $\mathcal{OTSE} = (\mathsf{E}, \mathsf{D})$ be a bijective one-time symmetric key encryption scheme. Let $\mathsf{A}^{\mathsf{PSA}}$ be an attacker modelled as a probabilistic Turing machine taking the security parameter k as input. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{PSA}}$ interacts with the "Challenger".

> **Phase 1**: Given a security parameter k, the Challenger picks key τ at random from SP_τ.
>
> **Phase 2**: $\mathsf{A}^{\mathsf{PSA}}$ chooses two equal-length plaintexts (m_0, m_1). If these are given to the encryption algorithm then the Challenger chooses $\beta \in \{0, 1\}$ at random, computes $\mathsf{E}_\tau(m_\beta)$, and gives $\mathsf{A}^{\mathsf{PSA}}$ the resulting target signcryptext c^*.
>
> **Phase 3**: $\mathsf{A}^{\mathsf{PSA}}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$.

We define $\mathsf{A}^{\mathsf{PSA}}$'s success by the probability

$$\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathsf{A}^{\mathsf{PSA}},\mathcal{OTSE}}(k) = 2\Pr[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathcal{SCR}}(t_{PSA})$ the maximum of the attacker $\mathsf{A}^{\mathsf{PSA}}$'s success over all attackers $\mathsf{A}^{\mathsf{PSA}}$ having running time t_{PSA}. Note that the running time and the number of queries are all polynomial in the security parameter k.

The $\mathcal{OTSE}$ scheme is said to be secure in the IND-PSA sense if $\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathcal{OTSE}}(t_{PSA})$ is negligible in k.

4.4.3 Confidentiality Proof

For confidentiality proof of Zheng's original signcryption scheme ZSCR, we adopt Shoup's methodology presented in [110]: We start with the real attack game where the attacker $\mathsf{A}^{\mathsf{CCA}}$ tries to defeat the security of the ZSCR scheme in the sense of FSO/FUO-IND-CCA defined in Section 4.2.2. We then modify this game by changing its rules and obtain a new game. Note here that the rules of each game are to describe how variables in the view of $\mathsf{A}^{\mathsf{CCA}}$ are computed. We repeat the modification until we obtain games related to the ability of the attackers $\mathsf{A}^{\mathsf{PSA}}$ and $\mathsf{A}^{\mathsf{gGDH}}$ to defeat the security of the one-time symmetric key encryption scheme $\mathcal{OTSE}$ and to solve the gGDH problem respectively. When a new game is derived from a previous one, a difference of the views of the attacker in each game might occur. This difference is measured by the technique presented in the following lemma.

Lemma 2 *Let* A_1, A_2, B_1 *and* B_2 *be events defined over some probability space. If* $\Pr[\mathsf{A}_1 \wedge \neg\mathsf{B}_1] = \Pr[\mathsf{A}_2 \wedge \neg\mathsf{B}_2]$ *and* $\Pr[\mathsf{B}_1] = \Pr[\mathsf{B}_2] = \varepsilon$ *then we have* $|\Pr[\mathsf{A}_1] - \Pr[\mathsf{A}_2]| \leq \varepsilon$.

The proof is a straightforward calculation and can be found in [110]. We now state and prove the following theorem.

Theorem 3 *If the gGDH problem is computationally intractable and the bijective one-time symmetric key encryption scheme* OTSE *used in Zheng's original signcryption scheme is IND-PSA secure then Zheng's original signcryption scheme* ZSCR *is FSO/FUO-IND-CCA secure in the random oracle model. Concretely, the following bound holds:*

$$\begin{aligned}\mathbf{Succ}^{\text{FSO/FUO}-\text{IND}-\text{CCA}}_{\mathtt{ZSCR}} \quad &(\quad t_{CCA}, q_{SC}, q_{USC}, q_G, q_H) \\ &\leq \quad 2\mathbf{Succ}^{\text{gGDH}}_{\mathbb{Z}_p^*}(t_{gGDH}, q_{O^{DDH}}) + \mathbf{Succ}^{\text{IND}-\text{PSA}}_{\mathtt{OTSE}}(t_{OTSE}) \\ &+ \quad q_{SC}\Big(\frac{q_G + q_H + 2q_{SC} + q_{USC} + 2}{2^{l_q(k)-1}}\Big) + \frac{q_H + 2q_{USC}}{2^{l_q(k)-1}},\end{aligned}$$

where $t_{gGDH} = t_{CCA} + O(q_G^2+1) + O(q_H^2+1) + O(k^3 q_{SC}) + O((k^3 + q_G + q_H)q_{USC}) + t^{OTSE}(q_{SC} + q_{USC})$, $q_{\mathsf{O}^{\mathsf{DDH}}} = (q_{SC} + q_{USC})(q_G + q_H)$ *and* $t_{OTSE} = O(t_{gGDH})$. *Note that* q_G *and* q_H*) denote the number of queries made by the FSO/FUO-IND-CCA attacker to the random oracles* G *and* H *used in* ZSCR*. Other parameters are as defined in the definitions of FSO/FUO-IND-CCA, IND-PSA, and the gGDH problem.*

Proof. Our aim is to keep modifying the real attack game of FUO/FSO-IND-CCA presented in Definition 7 until we get to the stage where one can defeat the IND-PSA of $\mathcal{OTSE}$ (Definition 12) and solve the gGDH problem (Definition 10).

We denote the FSO/FUO-IND-CCA attacker by "$\mathsf{A}^{\mathsf{CCA}}$" and use denote the attacker for the gGDH problem by "$\mathsf{A}^{\mathsf{gGDH}}$". Given (k, p, q, g, g^a, g^b) for random $a, b \in \mathbb{Z}_q^*$, $\mathsf{A}^{\mathsf{gGDH}}$'s goal is to compute the Diffie-Hellman key g^{ab} with the help of the DDH oracle $\mathcal{O}^{DDH}(\cdot,\cdot,\cdot,\cdot)$.

We start with the following game.

- Game $\mathbf{G_0}$: This game is the same as the real attack game presented in Definition 7.

  First, we run the common parameter/oracle generation algorithm GC of ZSCR on input a security parameter k and obtain a common parameter $cp =$

$(p, q, g, \mathsf{G}, \mathsf{H}, \mathtt{OTSE})$, where p and q are primes such that $|p| = k$, $q > 2^{l_q(k)}$, and $q|(p-1)$; g is an element in $\mathbb{Z}_p^*$ such that $\mathrm{Ord}_p(g) = q$; $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{l_G(k)}$ and $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ are hash functions modelled as the random oracles [20]; $\mathtt{OTSE}$ is the bijective one-time symmetric key encryption scheme that consists of the encryption function E and the decryption function D. We then run the key generation algorithm GK on input cp and k twice, and obtain Alice and Bob's (fixed) private/public key pairs. Here, Alice's private key consists of (x_A^*, y_A^*) where $y_A^* = g^{x_A^*}$, and her public key is y_A^* itself. Similarly, Bob's private key consists of (x_B^*, y_B^*) where $y_B^* = g^{x_B^*}$, and y_B^* itself is his public key.

We give the public key pair (y_A^*, y_B^*) to $\mathsf{A}^{\mathsf{CCA}}$. Once $\mathsf{A}^{\mathsf{CCA}}$ submits a pair of plaintexts (m_0, m_1) where $|m_0| = |m_1|$, we pick $\beta \in \{0,1\}$ uniformly at random and create a target signcryptext $C^* = (c^*, r^*, s^*)$ as follows.

$$c^* = \mathsf{E}_{\tau^*}(m_\beta);\ r^* = \mathsf{H}(m_\beta, y_A^*, y_B^*, K^*);\ s^* = x^*/(r^* + x_A^*),$$

where

$$K^* = {y_B^*}^{x^*}; \tau^* = \mathsf{G}(K^*)$$

for x^* picked uniformly at random from $\mathbb{Z}_q^*$. On input C^*, $\mathsf{A}^{\mathsf{CCA}}$ outputs $\beta' \in \{0,1\}$ at the end. We denote by S_0 the event $\beta' = \beta$ and use a similar notation S_i for all games $\mathbf{G}_i$.

Since this game is the same as the real attack game, we have

$$\Pr[S_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{FSO/FUO-IND-CCA}}_{\mathsf{A}^{\mathsf{CCA}},\mathtt{ZSCR}}(k).$$

- Game $\mathbf{G_1}$: In this game, we modify the target signcrytext C^* presented in the previous game. The modification obeys the following rules.

  R1-1 First, we choose $\tau^+ \in \{0,1\}^{l_G(k)}$, $r^+ \in \mathbb{Z}_q$, and $s^+ \in \mathbb{Z}_q^*$ uniformly at random. We then compute $c^+ = \mathsf{E}_{\tau^+}(m_\gamma)$ for random $\gamma \in \{0,1\}$ and replace c^*, r^*, s^*, and $\mathsf{G}(K^*)$ in the target signcryptext C^* by c^+, r^+, s^+, and τ^+ respectively. A new target signcryptext is now (c^+, r^+, s^+) and is denoted by C_+^*.

  R1-2 Whenever the random oracle G is queried at $K^* = (y_B^*)^{s^+(r^+ + x_A^*)}$ (as defined by r^+ and $s+$), we respond with τ^+.

  R1-3 Whenever the random oracle H is queried at $(m_\beta, y_A^*, y_B^*, K^*)$, where $K^* = (y_B^*)^{s^+(r^+ + x_A^*)}$, we respond with s^+.

R1-4 We assume that the signcryption and unsigncryption oracles are perfect. That is, on receiving $\mathsf{A}^{\mathsf{CCA}}$'s signcryption query (y_R, m) or unsigncryption query $(y_S, C) \neq (y_A^*, C^*)$, where y_S and y_R respectively denote sender and receiver's pubic keys arbitrarily selected by $\mathsf{A}^{\mathsf{CCA}}$, and m and C denote a message and a signcryptext respectively, we signcrypt (y_R, m) using the private key x_A^* or unsigncrypt (y_S, m) using the private key x_B^* in the same way as we do in the real attack game.

Since we have replaced one set of random variables by another set of random variables which is different, yet has the same distribution, the attacker $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_0}$ and Game $\mathbf{G_1}$ except for the event that $(m_\beta, y_A^*, y_B^*, K^*)$ is queried to H in Phase 3 (of the real attack game of FSO/FUO-IND-CCA) because we only know m_β at the end of Phase 3. The error is however small and is at most $q_H/2^{l_q(k)}$ because K^* is independent of the attacker's view in find stage.

Accordingly, we have

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{q_H}{2^{l_q(k)}}.$$

- Game $\mathbf{G_2}$: In this game, we retain the rules **R1-1** and **R1-4**, renaming them as "**R2-1**" and "**R2-4**" respectively. However, we drop the rules **R1-2** and **R1-3** meaning that τ^+ and s^+ are used only in producing the target signcryptext C_+^* while in other cases when the signcryption or unsigncryption oracle queries to the random oracles G and H, or $\mathsf{A}^{\mathsf{CCA}}$ directly queries to them, answers from G or H are taken. We refer to these rules regarding the random oracles G and H as "**R2-2**" and "**R2-3**" respectively.

  Since we have dropped the rule **R1-2**, τ^+ is not used anywhere in Game $\mathbf{G_2}$ except in computing c^*. Hence if $\beta' = \beta$ then $\mathsf{A}^{\mathsf{CCA}}$ has broken the IND-PSA security of the bijective one-time symmetric encryption scheme. Hence, we have

$$\Pr[S_2] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathsf{A}^{\mathsf{PSA}},\mathcal{OTSE}}(k).$$

  Now, let AskKey_2 denote an event that, in Game $\mathbf{G_2}$, G is queried at K^* by $\mathsf{A}^{\mathsf{CCA}}$ (rather than by the signcryption or unsigncryption oracles) or H is queried at (m, y', y'', K^*) for some (m, y', y'') by $\mathsf{A}^{\mathsf{CCA}}$ (again, rather than by the signcryption or unsigncryption oracles). We will use an identical notation AskKey_i for all the remaining games.

Notice that Game $\mathbf{G_1}$ and Game $\mathbf{G_2}$ may differ if G is queried at K^*, where $K^* = (y_B^*)^{s^+(r^+ + x_A^*)}$, or H is queried at $(m_\beta, y_A^*, y_B^*, K^*)$. Therefore, besides $\mathsf{AskKey_2}$, we need to consider the following events for which A's view may differ in Game $\mathbf{G_2}$.

- SCBad: G is queried at K^* or H is queried at $(m_\beta, y_A^*, y_B^*, K^*)$ by the signcryption oracle.
- USCBad: G is queried at K^* or H is queried at $(m_\beta, y_A^*, y_B^*, K^*)$ by the unsigncryption oracle.

But, SCBad has a negligible probability in Game $\mathbf{G_1}$: Namely due to the uniform distribution of K computed by the signcryption in the group $\langle g \rangle$, the probability that K hits K^* is less than $1/2^{l_q(k)}$ per each signcryption query. Consequently we have $\Pr[\mathsf{SCBad}] \leq q_{SC}/2^{l_q(k)}$.

We now show that $\mathsf{A}^{\mathsf{CCA}}$'s view is not changed from Game $\mathbf{G_1}$ to Game $\mathbf{G_2}$ except for a negligible subset of the USCBad event as well. For each unsigncryption query (y_S, c, r, s) processed by the unsigncryption oracle there are two cases to consider (Analysis of these events are given right after the description of each event.):

- *Case 1*: In processing (y_S, c, r, s), the unsigncryption oracle queries K^* to G and $(m_\beta, y_A^*, y_B^*, K^*)$ to H, but $\mathsf{A}^{\mathsf{CCA}}$ did not query $(m_\beta, y_A^*, y_B^*, K^*)$ to H.
  - In Game G_1, such unsigncryption queries (y_S, c, r, s) are always rejected by the unsigncryption oracle. The reason is that malling with the target signcryptext is impossible: if such a query (y_S, c, r, s) was accepted then it would have to be equal to the challenge (y_A^*, c^+, r^+, s^+), which is disallowed from being queried. To show this result, we note that by validity, we must have $r = r^+$. Then from equality of the H-queries we must have $y_S = y_A^*$ and $K^* = (y_B^*)^{s^+(r^+ + x_A^*)} = (y_B^*)^{s(r^+ + x_A^*)}$, which implies that also $s = s^+$ because y_B^* has order q. Finally, from equality of the H-queries we have $\mathsf{D}_{\tau^+}(c) = \mathsf{D}_{\tau^+}(c^+) = m_\beta$, which implies that also $c = c^+$, thanks to the one-to-one decryption function D of the *bijective* symmetric encryption scheme. Thus we conclude that $(y_S, c, r, s) = (y_A^*, c^+, r^+, s^+)$, so malling is impossible, as claimed.
  - In Game G_2, since $\mathsf{A}^{\mathsf{CCA}}$ did not query $(m_\beta, y_A^*, y_B^*, K^*)$ to H, we know that H was not previously queried at $(m_\beta, y_A^*, y_B^*, K^*)$ so H's response is independent of r^* and uniform in $\mathbb{Z}_q$, and hence the unsigncryption oracle accepts only with probability $1/2^{l_q(k)}$.

– *Case 2*: In processing (y_S, c, r, s), the unsigncryption oracle queries K^* to G and (m, y_S, y_B^*, K^*) to H for some for some $(m, y_S) \neq (m_\beta, y_A^*)$, but $\mathsf{A}^{\mathsf{CCA}}$ did not query (m, y_S, y_B^*, K^*) to H.

  * In Game G_1, since $\mathsf{A}^{\mathsf{CCA}}$ did not query (m, y_S, y_B^*, K^*) to H, we know that H was not previously queried at (m, y_S, y_B^*, K^*) so H's response is independent of r^+ and uniform in $\mathbb{Z}_q$, and hence the unsigncryption oracle accepts only with probability $1/2^{l_q(k)}$. If accepted, $\mathsf{A}^{\mathsf{CCA}}$ gets $\mathsf{D}_{\tau^+}(c)$.
  * In Game $\mathbf{G_2}$, since $\mathsf{A}^{\mathsf{CCA}}$ did not query (m, y_S, y_B^*, K^*) to H, we know that H was not previously queried at (m, y_S, y_B^*, K^*) so H's response is independent of r^+ and uniform in $\mathbb{Z}_q$, and hence the unsigncryption oracle accepts only with probability $1/2^{l_q(k)}$. If accepted, $\mathsf{A}^{\mathsf{CCA}}$ gets $\mathsf{D}_{\mathsf{G}(K^*)}(c)$.

  In either case 1 or case 2, the unsigncryption query (y_S, c, r, s) is answered differently in Game $\mathbf{G_2}$ compared to Game $\mathbf{G_1}$ for only a subset of outcomes in USCBad of probability at most $1/2^{l_q(k)}$, and hence at most $q_{USC}/2^{l_q(k)}$ over all unsigncryption queries.

Thus, finally we get

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[\mathsf{AskKey}_2] + \frac{q_{SC} + q_{USC}}{2^{l_q(k)}}.$$

- Game $\mathbf{G_3}$: We replace the rule **R2-1** in Game $\mathbf{G_2}$ by the following new rule **R3-1**.

  R3-1 First, we define K^* as $(y_B^*)^a$, τ^+ as $\mathsf{G}((y_B^*)^a)$, and s^+ as $\mathsf{H}(m_\beta, y_A^*, y_B^*, K^+)$. We then replace Bob's public key y_B^* by g^b and replace Alice's public key y_A^* by $(g^a g^{-r^+ s^+})^{\frac{1}{s^+}}$, where (g^a, g^b) for random $a, b \in \mathbb{Z}_q^*$ are the parameters given to the attacker $\mathsf{A}^{\mathsf{gGDH}}$. (Note that we get the same target signcryptext $C_+^* = (c^+, r^+, s^+)$ from this rule, but its components are redefined.)

However, we retain the rules **R2-2**, **R2-3** and **R2-4** in Game $\mathbf{G_2}$, renaming them as "**R3-2**", "**R3-3**" and "**R3-4**", respectively.

Thanks to the randomness of the oracle G and the uniformity of g^a and g^b in $\langle g \rangle$, $\mathsf{A}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_2}$ and Game $\mathbf{G_3}$. Hence, we have

$$\Pr[\mathsf{AskKey}_3] = \Pr[\mathsf{AskKey}_2].$$

- Game $\mathbf{G_4}$: In this game, we modify the rule **R3-4** and obtain a new rule **R4-4**. However, we retain the rules **R3-1**, **R3-2** and **R3-3** in Game $\mathbf{G_3}$, renaming them as "**R4-1**", "**R4-2**" and "**R4-3**" respectively.

  R4-4 In this rule, we replace the random oracles G and H by the random oracle simulators GSim and HSim. Note that two types of "query-answer" lists $\mathsf{GList1}$ and $\mathsf{GList2}$ are maintained for the simulation of the random oracle G. $\mathsf{GList1}$ consists of simple "input-output" entries for G of the form (K, τ). $\mathsf{GList2}$ (whose new entries are added by either the signcryption oracle simulator) consists of the special input-output entries for G which are of the form $y_R||\omega||(?, \tau)$. This implicitly represents the input-output relation $\tau = \mathsf{G}(\omega^{\log_g y_R})$, although the input $\omega^{\log_g y_R}$ is not explicitly stored and hence is denoted by "?". Similarly to GSim, the simulator HSim also maintains two input-output lists $\mathsf{HList1}$ and $\mathsf{HList2}$. $\mathsf{HList1}$ consists of simple input-output entries for H, which are of the form (μ, r). $\mathsf{HList2}$ (whose new entries are added by either the signcryption or unsigncryption oracle simulators in later games) consists of the special input-output entries for H which are of the form $y_R||\omega||((m, y_S, y_R, ?), r)$ and implicitly represents the input-output relation $\mathsf{H}(m, y_S, y_R, K) = r$, where $K = \omega^{\log_g y_R}$ is not explicitly stored and hence is denoted by "?". Complete specifications for GSim and HSim are as follows.

  Random Oracle Simulator GSim

  $\mathsf{GSim}(K)$
  - `If` $\mathcal{O}^{DDH}(g, \omega, y_R, K) = 1$
  - for some $y_R||\omega||(?, \tau) \in \mathsf{GList2}$
  - Return τ
  - Else if (K, τ) exists in $\mathsf{GList1}$
  - Return τ
  - Else $\tau \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^{l_G(k)}$
  - Return τ
  - Add (K, τ) to $\mathsf{GList1}$

Random Oracle Simulator HSim

$\mathsf{Hsim}(m, y_S, y_R, K)$
If $\mathcal{O}^{DDH}(g, \omega, y_R, K) = 1$ and
$y_R||\omega||(m, y_S, y_R, ?), r) \in \mathsf{HList2}$
Return r
Else if $((m, y_S, y_R, K), r)$
exists in $\mathsf{HList1}$ Return r
Else choose r at random from $\mathbb{Z}_q$
Return r
Add $((m, y_S, y_R, K), r)$ to $\mathsf{HList1}$

Notice that the above simulation for the random oracles G and H are perfect. Hence, we have

$$\Pr[\mathsf{AskKey_4}] = \Pr[\mathsf{AskKey_3}].$$

- Game $\mathbf{G_5}$: We retain all the rules **R4-1**, **R4-2** and **R4-3**, renaming them as "**R5-1**", "**R5-2**" and "**R5-3**" respectively. But we further modify **R4-4** and obtain a new rule "**R5-4**".

  R5-4 In this rule, we replace the signcryption oracle by the signcryption oracle simulator SCSim. On the other hand, we assume that the unsigncryption oracle is perfect.

Signcryption Oracle Simulator SCSim

$\mathsf{SCSim}(y_A^*, (y_R, m))$
If $y_R \notin \langle g \rangle$ Return "*Reject*"
Choose τ at random from $\{0,1\}^{l_G(k)}$
Choose r at random from $\mathbb{Z}_q$
Compute $c = \mathsf{E}_\tau(m)$
Choose s at random from $\mathbb{Z}_q^*$
Compute $\omega = (y_A^* g^r)^s$
Add $y_R||\omega||(?, \tau)$ to $\mathsf{GList2}$
Add $y_R||\omega||((m, y_A^*, y_R, ?), r)$ to $\mathsf{HList2}$
$C \leftarrow (c, r, s)$
Return C

Note that in the above signcryption oracle simulator,if neither (K, τ) nor $((m, y_A^*, y_R, K), r)$ exists in $\mathsf{GList1}$ and $\mathsf{HList1}$ respectively, the simulated signcryptext is distributed the same as the signcryptext in Game $\mathbf{G_4}$, but a simulation error occurs otherwise.

Thanks to the uniform distribution of K in the subgroup of $\mathbb{Z}_p^*$, whose order is q, and since $\mathsf{GList1}$ and $\mathsf{HList1}$ contain all queries to G and H both by the attacker,and the signcryption and unsigncryption oracles, we have $\Pr[(K, \tau) \in \mathsf{GList1}] \leq \frac{q_G + q_{SC} + q_{USC}}{2^{l_q(k)}}$ and $\Pr[((m, y_A^*, y_R, K), r) \in \mathsf{HList1}] \leq \frac{q_H + q_{SC} + q_{USC}}{2^{l_q(k)}}$. Also there is an error because in this game all signcryption queries are random but in the previous game, those with $r + x_A^*$ caused a "*Reject*" to be output by the signcryption oracle. Hence, we have $\Pr[r + x_A^* = 0] \leq \frac{1}{2^{l_q(k)}}$.

Since there are up to q_{SC} signcryption queries, the total probability of outcomes leading to signcryption oracle simulation error is upper-bounded by:

$$q_{SC}\Big(\frac{q_G + q_H + 2q_{SC} + q_{USC} + 1}{2^{l_q(k)}}\Big).$$

Summing up all decryption queries, we have

$$|\Pr[\mathsf{AskKey_5}] - \Pr[\mathsf{AskKey_4}]| \leq q_{SC}\Big(\frac{q_G + q_H + 2q_{SC} + q_{USC} + 1}{2^{l_q(k)}}\Big).$$

- Game $\mathbf{G_6}$: We retain all the rules **R5-1**, **R5-2** and **R5-3**, renaming them as "**R6-1**", "**R6-2**" and "**R6-3**" respectively. But we modify **R5-3** and obtain the following new rule "**R6-4**".

  > **R6-4** We replace the unsigncryption oracle by a unsigncryption oracle simulator USCSim which can unsigncrypt a submitted unsigncryption query (y_S, C) where $C = (c, r, s)$, without knowing the private key. Notice that the unsigncryption oracle simulator makes use of $\mathsf{A}^{\mathsf{gGDH}}$'s DDH oracle $\mathcal{O}^{DDH}$ to check whether a given tuple is Diffie-Hellman one.

Unsigncryption Oracle Simulator USCSim

$\mathsf{USC\,Sim}(y_B^*, y_S, C)$
If $y \notin \langle g \rangle$ Return "*Reject*"
Parse C as (c, r, s)
Compute $\omega = (y_S g^r)^s$
If $\omega = g^a$ Return "*Reject*"
If there exists $(K, \tau) \in \mathsf{GList1}$ such that
$\mathcal{O}^{DDH}(g, \omega, y_B^*, K) = 1$ or
there exists $y_R||\omega'||(?, \tau) \in \mathsf{GList2}$ such that
$\mathcal{O}^{DDH}(\omega, \omega', y_R, y_B^*) = 1$
  Compute $m = \mathsf{D}_\tau(c)$
Else Choose τ at random from $\{0,1\}^{l_G(k)}$;
  Add $y_B^*||\omega||(?, \tau)$ to $\mathsf{Glist2}$;
  Compute $m = \mathsf{D}_\tau(c)$
If there exists $((m, y_S, y_B^*, K), r) \in \mathsf{HList1}$ such that
$\mathcal{O}^{DDH}(g, \omega, y_B^*, K) = 1$ or
there exists $y_R||\omega'||((m, y_S, y_R, ?), r) \in \mathsf{HList2}$ such that
$\mathcal{O}^{DDH}(\omega, \omega', y_R, y_B^*) = 1$
  Return m
Else Return "*Reject*"

In the above simulation, if $K(= (y_S g^r)^{s x_B^*})$ has not been queried to G and (m, y_S, y_B^*, K) has not been queried to H but C is a valid signcryptext then a difference between Game $\mathbf{G_5}$ and Game $\mathbf{G_6}$ occurs: If $C = (c, r, s)$ is valid, we have $m = \mathsf{D}_{\mathsf{G}(K)}(c)$ where $K = (y_S g^r)^s$, and $r = \mathsf{H}(m, y_S, y_B^*, K)$. Since we have assumed that $\mathsf{G}(K)$ or $\mathsf{H}(m, y_S, y_B^*, K)$ has not been queried, the event $r = \mathsf{H}(m, y_S, y_B^*, K)$ happens with probability $1/2^{l_q(k)}$ since the output of the random oracle H is uniformly distributed in $\mathbb{Z}_q$.

Summing up over all the unsigncryption queries, we have

$$|\Pr[\mathsf{AskKey_6}] - \Pr[\mathsf{AskKey_5}]| \leq \frac{q_{USC}}{2^{l_q(k)}}.$$

Since Game $\mathbf{G_3}$, $\mathsf{AskKey_6}$ has denoted the event that the Diffie-Hellman key $(y_B^*)^a (= g^{ab})$ has been queried to the random oracle G or H. At this stage, we can check which one of the queries to the random oracles G and H is a Diffie-Hellman key of g^{ab} using $\mathsf{A}^{\mathsf{gGDH}}$'s DDH oracle $\mathcal{O}^{DDH}$. Furthermore, note that we have used the DDH oracle to simulate the unsigncryption oracle. That is, we have

now reached the stage where we can solve the gGDH problem and hence we have

$$\Pr[\mathsf{AskKey}_6] \leq \mathbf{Succ}^{\mathrm{gGDH}}_{\mathbb{Z}_p^*,\mathsf{A}^{\mathsf{gGDH}}}(k).$$

Putting all the bounds we have obtained in each game together, we obtain

$$\begin{aligned}
\frac{1}{2}\mathbf{Succ}^{\mathrm{FSO/FUO-IND-CCA}}_{\mathsf{A}^{\mathsf{CCA}},\mathtt{ZSCR}}(k) &= |\Pr[S_0] - \Pr[S_2]| \\
&\leq \frac{q_H}{2^{l_q(k)}} + \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathsf{A}^{\mathsf{PSA}},\mathtt{OTSE}}(l) + \frac{q_{SC}+q_{USC}}{2^{l_q(k)}} \\
&+ q_{SC}\Big(\frac{q_G+q_H+2q_{SC}+q_{USC}+1}{2^{l_q(k)}}\Big) \\
&+ \frac{q_{USC}}{2^{l_q(k)}} + \Pr[\mathsf{AskKey}_6] \\
&\leq \frac{1}{2}\mathbf{Succ}^{\mathrm{IND-PSA}}_{\mathsf{A}^{\mathsf{PSA}},\mathcal{OTSE}}(l) \\
&+ q_{SC}\Big(\frac{q_G+q_H+2q_{SC}+q_{USC}+2}{2^{l_q(k)}}\Big) \\
&+ \frac{q_H+2q_{USC}}{2^{l_q(k)}} + \mathbf{Succ}^{\mathrm{gGDH}}_{\mathbb{Z}_p^*,\mathsf{A}^{\mathsf{gGDH}}}(k).
\end{aligned}$$

The advantage bound claim of the theorem follows upon taking maximums over all attackers with the appropriate resource parameters. The running time counts can be readily checked.

□

4.4.4 Unforgeability Proof

We now show that Zheng's original signcryption scheme ZSCR is FiSO-UF-CMA (Definition 8) secure.

As a proof methodology, we use Ohta and Okamoto's [88] "ID reduction technique": We first convert a forger that conducts chosen message attack against ZSCR to an attacker that conducts *passive* attack against an identification scheme derived from ZSCR. We then use this attacker to construct a matrix which contains a heavy row which will be defined later. Using the heavy row, we can construct an attacker that finds a discrete-logarithm of the Alice's public key with non-negligible probability.

Now, we present the following identification scheme IZSCR derived from ZSCR.

Suppose that a common parameter $cp = (k, p, q, g)$, where k is a security parameter; p and q are random primes such that $|p| = k$, $q > 2^{l_q(k)}$, and $q|(p-1)$ ($l_q : \mathbb{N} \to \mathbb{N}$ is a function determining the length of q); g is an element of $\mathbb{Z}_p^*$ such that $\mathrm{Ord}_p(g) = q$, is given to the Prover and the Verifier.

The Prover then chooses x_P at random from $\mathbb{Z}_q$, computes $y_P = g^{x_P}$, and publishes y_P. Similarly, The Verifier randomly chooses x_V from $\mathbb{Z}_q$ and computes $y_V = g^{x_V}$, and publishes y_V. Now the Prover and the Verifier perform the following protocols.

Step 1: The Prover randomly chooses x from $\mathbb{Z}_q$, computes $K = y_V^x$ and sends K to the Verifier.

Step 2: On receiving K from the Prover, the Verifier randomly chooses r from $\mathbb{Z}_q$ and sends r to the Prover.

Step 3: The Prover computes $s = x/(r + x_P) \in \mathbb{Z}_q$ and sends s to V.

Step 4: The Verifier computes $K' = (y_P g^r)^{s x_V}$ and returns "*Accept*" if $K' = K$, otherwise, returns "*Reject*".

Now we define a security notion for IDZSCR similar to the one defined in [88]. We call our notion "SI-PSV (security of an identification scheme against passive attack)".

Definition 13 (SI-PSV) Let IDZSCR be an identification scheme derived from Zheng's original signcryption scheme. Let $\mathsf{A}^{\mathsf{IDPSV}}$ be an attacker modelled as a probabilistic Turing machine modelled as a probabilistic Turing machine taking the security parameter k as input. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{IDPSA}}$ interacts with the "Challenger" that controls the Prover and the Verifier.

Phase 1: The Challenger runs the common parameter generation algorithm $\mathsf{GC}(k)$ of IDZSCR, and obtains a common parameter $cp = (k, p, q, g)$. The Challenger then runs the user key-pair generation algorithm $\mathsf{GK}(cp)$ twice to generate private/public key pairs (x_P^*, y_P^*) and (x_V^*, y_V^*), where $y_P^* = g^{x_P^*}$ and $y_V^* = g^{x_V^*}$. The Challenger gives cp and (x_P^*, y_P^*) to the Prover, and gives cp and (x_V^*, y_V^*) to the Verifier. Finally, the Challenger gives cp, y_P^*, and (x_V^*, y_V^*) to $\mathsf{A}^{\mathsf{IDPSV}}$.

Phase 2: Given $(cp, y_P^*, x_V^*, y_V^*)$, $\mathsf{A}^{\mathsf{IDPSV}}$ acts as a Prover as follows. In Step 1 of IDZSCR, $\mathsf{A}^{\mathsf{IDPSV}}$ sends arbitrary value drawn from $\mathbb{Z}_p^*$ to the Challenger. The Challenger then forwards this value as a message in Step 1 of IDZSCR to the Verifier. On receiving a response to this message from the Verifier, the Challenger just sends that response back to $\mathsf{A}^{\mathsf{IDPSV}}$. In Step 3 of IDZSCR, $\mathsf{A}^{\mathsf{IDPSV}}$ sends arbitrary value drawn from $\mathbb{Z}_p^*$ to the Challenger. The Challenger then forwards this value as a message in Step 3 of IDZSCR to the Verifier.

We define $\mathsf{A}^{\mathsf{IDPSV}}$'s success by the probability

$$\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k) = \Pr[\text{The Verifier outputs } \text{“}\textit{Accept}\text{”}].$$

We denote by $\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathtt{IDZSCR}}(t_{IDPSV})$ the maximum of the attacker $\mathsf{A}^{\mathsf{IDPSV}}$'s success over all attackers $\mathsf{A}^{\mathsf{IDPSV}}$ having running time t_{IDPSV}. Note that the running time and the number of queries are all polynomial in the security parameter k.

The IDZSCR scheme is said to be secure in the SI-PSV sense if $\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathtt{IDZSCR}}(t_{IDPSV})$ is negligible in k.

We now show that the scheme ZSCR is FiSO-UF-CMA secure assuming that the DL problem (Definition 11) is computationally intractable.

As a first step, we reduce the SI-PSV security of IDZSCR to the FiSO-UF-CMA security of ZSCR.

Lemma 3 *Suppose that an FiSO-UF-CMA attacker for the scheme* ZSCR*, whose running time is bounded by t_{CMA}, issues up to q_{SC} queries to the signcryption oracle, q_G and q_H to the random oracles* G *and* H *respectively. Using this attacker as a subroutine, we can construct a SI-PSV attacker for the scheme* IDZSCR*, whose running time is bounded by t_{IDPSV}. Concretely, we obtain the following advantage bound:*

$$\frac{1}{q_H}\Big(\mathbf{Succ}^{\mathrm{FiSO-UF-CMA}}_{\mathtt{ZSCR}}(t_{CMA}, q_{SC}, q_G, q_H) - \frac{1}{2^{l_q(k)}}\Big) \;\leq\; \mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathtt{IDZSCR}}(t_{IDPSV}),$$

where $t_{IDPSV} = t_{CMA} + O(q_{SC} + q_G + q_H) + O(k^3)$ for a security parameter k.

Proof. Let $\mathsf{A}^{\mathsf{CMA}}$ denote an attacker (forger) that defeats the FiSO-UF-CMA security of the ZSCR scheme. Assume that the common parameter $cp = (k, p, q,$

g, G, H, $\mathtt{OTSE}$) of $\mathtt{ZSCR}$, Alice's public key y_A^* and Bob's public and private key pairs (y_B^*, x_B^*) are provided as input to $\mathsf{A}^{\mathsf{CMA}}$.

Our aim is to simulate the view of $\mathsf{A}^{\mathsf{CMA}}$ in the real attack game denoted by $\mathbf{G_0}$ until we obtain a game in which the attacker $\mathsf{A}^{\mathsf{IDPSV}}$ for the $\mathtt{IDZSCR}$ scheme defeats the SI-PSV security of $\mathtt{IDZSCR}$.

- Game $\mathbf{G_0}$: As mentioned, this game is identical to the real attack game described in Definition 8. We denote by S_0 the event that $\mathsf{A}^{\mathsf{CMA}}$ succeeds in forging a signcryptext and use a similar notation S_i for all games $\mathbf{G}_i$. Since Game $\mathbf{G_0}$ is the same as the real attack game, we have

$$\Pr[S_0] = \mathbf{Succ}^{\mathrm{FiSO-UF-CMA}}_{\mathtt{ZSCR},\mathsf{A}^{\mathsf{CMA}}}(k).$$

- Game $\mathbf{G_1}$: We modify the previous game $\mathbf{G_0}$ and obtain a new game $\mathbf{G_1}$ as follows:

  We respond to $\mathsf{A}^{\mathsf{CMA}}$'s queries in accordance with the following rules.

  R1-1 If $\mathsf{A}^{\mathsf{CMA}}$ makes a new query K_u to the random oracle G, we choose τ_u uniformly at random from $\{0,1\}^{l_G(k)}$ and respond with τ_u. We then update the entry of "query-answer" list GList with $\langle K_u, \tau_u \rangle$. If the same query K_u is asked, we search for the corresponding value τ_u in GList and respond with it.

  R1-2 We choose j uniformly at random from $[1, \ldots, q_H]$, where q_H denotes the maximum number of queries to the random oracle H. If $\mathsf{A}^{\mathsf{CMA}}$ makes a query μ_j to the random oracle H, we respond with $\mathsf{H}(\mu_j)$. On the other hand, if $\mathsf{A}^{\mathsf{CMA}}$ makes a new query μ_i where $i \neq j$ to the random oracle H, then we choose r_i uniformly at random from $\mathbb{Z}_q$ and respond with r_i. We then update the entry of "query-answer" list HList with $\langle \mu_i, r_i \rangle$. If the same query μ_i is asked, we search for the corresponding answer for r_i in HList and respond with it.

  R1-3 If $\mathsf{A}^{\mathsf{CMA}}$ makes a query (y_B^*, m_v) to its signcryption oracle, then we run $\mathsf{SC}(cp, x_A^*, y_B^*, m_v)$ and give the resulting output to $\mathsf{A}^{\mathsf{CMA}}$.

  R1-4 Once $\mathsf{A}^{\mathsf{CMA}}$ submits a forgery (m^*, C^*) where $C^* = (c^*, r^*, s^*)$, we check if there exists $\langle K_{u'}, \tau_{u'} \rangle$, where $u' \in [1, \ldots, q_G]$ (q_G denotes the number of queries to the random oracle G), in the entries in GList such that $(y_A^* g^{r^*})^{s^* x_B^*} = K_{u'}$. If there exists such an entry, we set $K^* = K_{u'}$. Otherwise, we choose K^* at random from $\mathbb{Z}_q^*$.

Having obtained K^*, we check if $(m^*, y_A^*, y_B^*, K^*) = \mu_j$ and $r^* = \mathsf{H}(\mu_j)$. If this test holds, we return (m^*, C^*), otherwise, we return "*Abort*" and terminate the game.

Now, we define the following event.

– AskH: There exists $i \in [1, \ldots, q_H]$ such that $(m, y_A^*, y_B^*, K) = \mu_i$ for some $m \in SP_m$ and random $K \in \mathbb{Z}_q^*$. (Or equivalently, (m, y_A^*, y_B^*, K) has been asked to the random oracle H).

From the rules defined above, we obtain

$$\begin{aligned}
\Pr[S_1] &\geq \sum_{i=1}^{q_H} \Pr[(i = j) \wedge \mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*) \wedge \mathsf{AskH}] \\
&= \sum_{i=1}^{q_H} \Pr[i = j] \Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*) \wedge \mathsf{AskH}] \\
&= \Pr[i = j] \sum_{i=1}^{q_H} \Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*) \wedge \mathsf{AskH}] \\
&= \frac{1}{q_H}\Big(\Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*)] \\
&- \Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*) \wedge \neg\mathsf{AskH}]\Big) \\
&= \frac{1}{q_H}\Big(\Pr[S_0] - \frac{1}{2^{l_q(k)}}\Big)
\end{aligned}$$

Note that $\Pr[\mathsf{A}^{\mathsf{CMA}} \text{ outputs } (m^*, C^*) \wedge \neg\mathsf{AskH}] = \frac{1}{2^{l_q(k)}}$ due to the uniformity of outputs of the random oracle H.

- Game $\mathbf{G_2}$: In this game, we retain the rules **R1-1**, **R1-2**, and **R1-4**, renaming them as "**R2-1**", "**R2-2**", and "**R2-4**" respectively. But we modify **R1-3** and obtain a new rule "**R2-3**".

  R2-3 If $\mathsf{A}^{\mathsf{CMA}}$ makes a new query (y_B^*, m_i) to its signcryption oracle, then we search HList for an entry $\langle \mu_i, r_i \rangle$ such that $\mu_i = (m_i, y_A^*, y_B^*, K_i)$ for some $K_i \in \mathbb{Z}_q^*$. If there exists one, we do the following.

  * Search GList for an entry $\langle K_i, \tau_i \rangle$. If there exists one, compute $c_i = \mathsf{E}_{\tau_i}(m_i)$. Otherwise, choose τ_i at random from $\{0,1\}^{l_G(k)}$ and compute $c_i = \mathsf{E}_{\tau_i}(m_i)$.
  * Choose s_i at random from $\mathbb{Z}_q$.

If there does not exist $\langle \mu_i, r_i \rangle$ such that $\mu_i = (m_i, y_A^*, y_B^*, K_i)$ in HList, we do the following.

* Choose r_i at random from $\mathbb{Z}_q$ and put $\langle \mu_i, r_i \rangle$ into HList.
* Extract K_i from μ_i. Then, search GList for an entry $\langle K_i, \tau_i \rangle$. If there exists one, compute $c_i = \mathsf{E}_{\tau_i}(m_i)$. Otherwise, choose τ_i at random from $\{0,1\}^{l_G(k)}$ and compute $c_i = \mathsf{E}_{\tau_i}(m_i)$.
* Choose s_i at random from $\mathbb{Z}_q$.

Note that either case, we obtain $C_i = (c_i, r_i, s_i)$. We then return this as an answer for the query (y_B^*, m_i). Finally, we put $\langle (y_B^*, m_i), C_i \rangle$ into a "query-answer" list SCList for the signcryption oracle SC. Of course, the same query (y_B^*, m_i) is asked to the signcryption oracle twice, we search SCList for the corresponding entry $\langle (y_B^*, m_i), C_i \rangle$ and answer with C_i.

Note that the above rule makes no change in $\mathsf{A}^{\mathsf{CMA}}$'s view in Game $\mathbf{G_1}$. Hence, we get

$$\Pr[S_2] = \Pr[S_1].$$

- Game $\mathbf{G_3}$: In this game, we further modify the rules of the previous game to obtain a game in which $\mathsf{A}^{\mathsf{IDPSV}}$ conducts a passive attack on the `IDZSCR` scheme.

  We now describe the modification as follows: We retain the rules **R2-1** and **R2-3**, renaming them as "**R3-1**", "**R3-2**" respectively. But we modify **R2-2** and **R2-4** obtain new rules "**R3-2**" and "**R3-4**" respectively.

  R3-2 We choose j uniformly at random from $[1, \ldots, q_H]$. If $\mathsf{A}^{\mathsf{CMA}}$ makes a query μ_j to the random oracle H, then we perform the following.

  * Parse μ_j as (m_j, y_A^*, y_B^*, K_j).
  * Send K_j to the Verifier of `IDZSCR`.
  * When the Verifier makes a random challenge $\tilde{r} \in \mathbb{Z}_q$, forward it to $\mathsf{A}^{\mathsf{CMA}}$ as an answer to its query μ_j (to the random oracle H).

  On the other hand, if $\mathsf{A}^{\mathsf{CMA}}$ makes a new query μ_i where $i \neq j$ to the random oracle H, then we choose r_i uniformly at random from $\mathbb{Z}_q$ and respond with r_i. We then update the entry of "query-answer" list HList with $\langle \mu_i, r_i \rangle$. If the same query μ_i is asked, we search for the corresponding value r_i in HList and respond with it.

R3-4 Once $\mathsf{A}^{\mathsf{CMA}}$ submits a forgery (m^*, C^*) where $C^* = (c^*, r^*, s^*)$, we check if there exists $\langle K_{u'}, \tau_{u'} \rangle$, where $u' \in [1, \ldots, q_G]$, in the entries in GList such that $(y_A^* g^{r^*})^{s^* x_B^*} = K_{u'}$. If there exists such entry, we set $K^* = K_{u'}$. Otherwise, we choose K^* at random from $\mathbb{Z}_q^*$.
Having obtained K^*, we check if $(m^*, y_A^*, y_B^*, K^*) = \mu_j$ and $r^* = \tilde{r}$. If this test holds, we send s^* to the Verifier of IDZSCR, otherwise, we return "*Abort*" and terminate the game.

If $\mathsf{A}^{\mathsf{CMA}}$ outputs a valid signcryptext, then the Verifier accepts $\mathsf{A}^{\mathsf{IDPSV}}$'s proof. Hence, we have

$$\Pr[S_3] = \mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k).$$

Putting all the bounds we have obtained in each game together, we obtain

$$\frac{1}{q_H}\Big(\mathbf{Succ}^{\mathrm{FiSO-UF-CMA}}_{\mathtt{ZSCR},\mathsf{A}^{\mathsf{CMA}}}(k) - \frac{1}{2^{l_q(k)}}\Big) \;\leq\; \mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k).$$

□

We now proceed to show that one can construct an attacker that finds discrete-logarithm of Alice's public key $y_A^* = g^{x_A^*}$ using $\mathsf{A}^{\mathsf{IDPSV}}$ as a subroutine. We first define a "Heavy Row" as follows.

Definition 14 (Heavy Row) Assume that there is an attacker $\mathsf{A}^{\mathsf{IDPSV}}$ that conducts the passive attack against the IDZSCR scheme with $\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k) \geq \frac{4}{2^{l_q(k)}}$, where k and $l_q(k)$ are as defined in the description of the ZSCR scheme. Let $\Phi(RA, r)$ be a boolean matrix whose rows and columns correspond to $\mathsf{A}^{\mathsf{IDPSV}}$'s private random strings and the Verifier's all possible choices of random challenge r respectively. Its entries are 0 if the Verifier rejects $\mathsf{A}^{\mathsf{IDPSV}}$'s proof, and 1 if the Verifier accepts $\mathsf{A}^{\mathsf{IDPSV}}$'s proof. A row of $\Phi(RA, r)$ is said to be *heavy* if the fraction of 1's along the row is at least $\frac{1}{2}\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k)$.

We now state and prove the following lemma called the "Heavy Row Lemma" [88].

Lemma 4 *The 1's in $\Phi(RA, r)$ are located in heavy rows of $\Phi(RA, r)$ with a probability of at least $\frac{1}{2}$.*

Proof. By definition of heavy row, the fraction of 1's in a row which corresponds to one column is at least $\frac{1}{2}\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k)$.

Now, let $\mathsf{LocHeavyRow}$ denote an event that the 1's in $\Phi(RA, r)$ are located in heavy rows of $\Phi(RA, r)$. Since "1" appears $1/\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k)$ times in $\Phi(RA, r)$, we have

$$\Pr[\mathsf{LocHeavyRow}] \geq \frac{1}{\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k)} \cdot \frac{1}{2}\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k) = \frac{1}{2}.$$

□

Now, suppose that a FiSO-UF-CMA attacker for ZSCR succeeds in forging signcryptext with the probability greater than $(4q_H+1)/2^{l_q(k)}$, where q_H is the number of queries to the random oracle H. Then, by Lemma 3, we have

$$\begin{aligned}\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathtt{IDZSCR}}(t_{IDPSV}) &\geq \frac{1}{q_H}\Big(\mathbf{Succ}^{\mathrm{FiSO-UF-CMA}}_{\mathtt{ZSCR}}(t_{CMA}, q_{SC}, q_G, q_H) - \frac{1}{2^{l_q(k)}}\Big) \\ &\geq \frac{4}{2^{l_q(k)}}.\end{aligned}$$

It just remains to show that using the SI-PSV attacker for IDZSCR, whose success probability is at least $4/2^{l_q(k)}$, one can construct an attacker that finds the discrete-logarithm $x^* \in \mathbb{Z}_q^*$ of Alice's public key $y^* = g^{x^*} \in \mathbb{Z}_p^*$ with non-negligible probability.

Lemma 5 *Using a SI-PSV attacker for the scheme* IDZSCR*, whose success probability is at least (greater than or equal to)* $4/2^{l_q(k)}$ *and running time is bounded by* t_{IDPSV}*, as a subroutine, we can construct an attacker that finds the discrete-logarithm* $x^* \in \mathbb{Z}_q^*$ *of* $g^{x^*} \in \mathbb{Z}_p^*$*. Concretely, we obtain the following bound:*

$$\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathtt{IDZSCR}}(t_{IDPSV}) \leq 2\sqrt{\mathbf{Succ}^{\mathrm{DL}}_{\mathbb{Z}_p^*}(t_{DL})},$$

where $t_{DL} = O(t_{IDPSV}) + O(k^3)$.

Proof. Let $\mathsf{A}^{\mathsf{IDPSV}}$ denote an attacker that defeats the SI-PSV security of the IDZSCR scheme. Assume that the common parameter $cp = (k, p, q, g)$ of IDZSCR, the Prover's public key y_P^* and the Verifier's public/private key pair (y_V^*, x_V^*) are provided as input to $\mathsf{A}^{\mathsf{IDPSV}}$.

Our aim is to simulate the view of $\mathsf{A}^{\mathsf{IDPSV}}$ in the real attack game denoted by $\mathbf{G_0}$ to obtain a new game $\mathbf{G_1}$ in which the attacker A^{DL} finds the discrete-logarithm $x^* \in \mathbb{Z}_q^*$ of $y^* = g^{x^*} \in \mathbb{Z}_p^*$.

- Game $\mathbf{G_0}$: As mentioned, this game is the real attack game in which an attacker $\mathsf{A}^{\mathsf{IDPSV}}$ attacks the scheme IDZSCR in the SI-PSV sense. We denote by S_0 the event that $\mathsf{A}^{\mathsf{IDZSCR}}$ succeeds in masquerading itself as the Prover of IDZSCR and making the Verifier accept $\mathsf{A}^{\mathsf{IDZSCR}}$'s outputs. Since Game $\mathbf{G_0}$ is the same as the real attack game, we have

$$\Pr[S_0] = \mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k).$$

- Game $\mathbf{G_1}$: In this game, we modify the previous game as follows.
  - Replace the Prover's public key y_P^* that $\mathsf{A}^{\mathsf{IDPSV}}$ has access to by $y^* = g^{x^*}$ which A^{DL} tries to find its discrete-logarithm.
  - Form $\Phi(RA, r)$ as $\mathsf{A}^{\mathsf{IDPSV}}$ attacks IDZSCR in SI-PSV sense. More precisely, we record all the (K, r, s)'s where K, r, and s are the values exchanged from $\mathsf{A}^{\mathsf{IDPSV}}$ and the Verifier and give 1 if the Verifier outputs "*Accept*" and give 0, otherwise.
  - Search for an entry $\vec{a_0}(= (K_0, r_0, s_0))$ with 1 in $\Phi(RA, r)$. By $\Phi_0(RA, r)$, denote the entry $\vec{a_0}$ is located.
  - Search for an entry $\vec{a_1}(= (K_1, r_0, s_0))$ with 1 in $\Phi_0(RA, r)$.
  - Compute $x^* = \frac{s_0 r_0 - s_1 r_1}{s_0 - s_1} \in \mathbb{Z}_q^*$ and return x^* as A^{DL}'s answer.

  Note that for $\vec{a_0} = (K_0, r_0, s_0)$, $K_0 = (y_P^* g^{r_0})^{s_0 x_V^*} = (y^* g^{r_0})^{s_0 x_V^*}$. Similarly, for $\vec{a_1} = (K_1, r_1, s_1)$, $K_1 = (y_P^* g^{r_1})^{s_1 x_V^*} = (y^* g^{r_1})^{s_1 x_V^*}$. Since $\vec{a_0}$ and $\vec{a_1}$ are the entries with 1, both of which are in $\Phi_0(RA, r)$, we have $K_0 = K_1$. Thus we can obtain the discrete-logarithm x^* of $y^*(= g^{x^*})$ by computing,

$$x^* = \frac{s_0 r_0 - s_1 r_1}{s_0 - s_1} \in \mathbb{Z}_q^*.$$

  Now, let $\mathsf{E_0}$ denote an event that the entry $\vec{a_0}$ is found in $\Phi(RA, r)$. Since this happens whenever $\mathsf{A}^{\mathsf{IDPSV}}$ succeeds, we have

$$\Pr[\mathsf{E_0}] \geq \Pr[S_0].$$

  We then denote by $\mathsf{E_1}$ an event that the entry $\vec{a_1}$ is found in $\Phi_0(RA, r)$. By Lemma 14 (heavy row lemma) and the fact that the fraction of 1's along a heavy row is at least $\frac{1}{2}\Pr[S_0]$, we have

$$\Pr[\mathsf{E_1}] \geq \frac{1}{2} \cdot \frac{1}{2} \Pr[S_0] = \frac{1}{4} \Pr[S_0].$$

Hence, if we denote by S_1 the event that A^{DL} outputs x^* in this game, we obtain

$$\mathbf{Succ}^{\mathrm{DL}}_{\mathsf{A}^{\mathsf{DL}},\mathbb{Z}_p^*}(k) = \Pr[\mathsf{E}_0]\Pr[\mathsf{E}_1] \geq \frac{1}{4}\Pr[S_0]^2 = \frac{1}{4}(\mathbf{Succ}^{\mathrm{SI-PSV}}_{\mathsf{A}^{\mathsf{IDPSV}},\mathtt{IDZSCR}}(k))^2.$$

Considering the running time, we obtain the bound in the lemma statement. □

Combining the results of Lemmas 3 and 5, we finally obtain the following theorem.

Theorem 4 *If the DL problem is computationally intractable, then Zheng's original signcryption scheme* ZSCR *is FiSO-UF-CMA secure in the random oracle model. Concretely, for a FiSO-UF-CMA attacker for* ZSCR *succeeds in forging signcryptext with the probability greater than* $(4q_H+1)/2^{l_q(k)}$*, the following bound holds:*

$$\frac{1}{q_H}\mathbf{Succ}^{\mathrm{FiSO-UF-CMA}}_{\mathtt{ZSCR}}(t_{CMA}, q_{SC}, q_G, q_H) \leq 2\sqrt{\mathbf{Succ}^{\mathrm{DL}}_{\mathbb{Z}_p^*}(t_{DL})} + \frac{1}{q_H 2^{l_q(k)}},$$

where q_{SC} *is the number of queries to the signcryption oracle* SC*;* q_G *and* q_H *are the number of queries to the random oracles* G *and* H*;* $t_{DL} = O(t_{CMA} + q_{SC} + q_G + q_H) + O(k^3)$*.*

4.5 Brief Summary of the Results

In this chapter, we extensively discussed issues related to the integration of message confidentiality and authenticity in the symmetric and public key settings. We then proved that Zheng's original signcryption scheme is "FSO/FUO-IND-CCA" secure in the random oracle model, relative to the (generalized) GDH problem. Although our FSO/FUO-IND-CCA notion is similar to the well known "IND-CCA" notion defined for standard public key encryption schemes, it is stronger than the direct adaptation of IND-CCA to the setting of signcryption, since we allow an attacker to query both the signcryption oracle and the unsigncryption oracle in a flexible way. We also proved that Zheng's 's original signcryption scheme satisfies our strong unforgeability notion called "FiSO-UF-CMA" in the random oracle model assuming that the standard DL problem is computationally intractable.

Chapter 5

Identity-Based Threshold Decryption from the Bilinear Map

5.1 Introduction

5.1.1 Motivation

Threshold decryption is particularly useful where the centralization of the power to decrypt is a concern. And the motivation for identity (ID)-based encryption originally proposed by Shamir [108] is to provide confidentiality without the need of exchanging public keys or keeping public key directories. A major advantage of ID-based encryption is that it allows one to encrypt a message by using a recipient's identifiers such as an email address.

A combination of these two concepts will allow one to build an "ID-based threshold decryption" scheme. One possible application of such a scheme can be considered in a situation where an identity denotes the name of the group sharing a decryption key. As an example, suppose that Alice wishes to send a confidential message to a committee in an organization. Alice can first encrypt the message using the identity (name) of the committee and then send over the ciphertext. Let us assume that Bob who is the committee's president has created the identity and hence has obtained a matching private decryption key from the Private Key Generator (PKG). Preparing for the time when Bob is away, he can share his private key out among a number of decryption servers in such a way that

any committee member can successfully decrypt the ciphertext if, and only if, the committee member obtains a certain number of decryption shares from the decryption servers.

Another application of the ID-based threshold decryption scheme is to use it as a building block to construct a mediated ID-based encryption scheme [46]. The idea is to split a private key associated with the receiver Bob's ID into two parts, and give one share to Bob and the other to the Security Mediator (SEM). Accordingly, Bob can decrypt a ciphertext only with the help of the SEM. As a result, instantaneous revocation of Bob's privilege to perform decryption is possible by instructing the SEM not to help him any more.

5.1.2 Contributions of This Chapter

The contributions of this chapters are twofold. First, we construct an ID-based threshold decryption scheme which can be well-applicable to the situations described in the previous section. Our scheme has two advantages over the schemes appeared in the literature previously, which will be discussed in detail in Section 5.3: 1) Our scheme meets a strong security requirement, that is, our schemes gives *chosen ciphertext security* which is proven in the random oracle model [20], assuming the Bilinear Diffie-Hellman problem [27] is computationally intractable; 2) Our scheme has a useful feature that the private key associated with an ID rather than the master key of the PKG is shared among decryption servers. Second, we apply our ID-based threshold decryption scheme to design a mediated ID-based encryption scheme secure against more powerful attacks than those considered previously in the literature.

5.2 Preliminaries

5.2.1 The Admissible Bilinear Map

To begin with, we review the admissible bilinear map which payed the central role in Boneh and Franklin's ID-based encryption scheme [27].

The admissible bilinear map, which we denote by $\hat{e}$, is defined over two groups of the same prime-order q denoted by $\mathcal{G}$ and $\mathcal{F}$ in which the Computational Diffie-Hellman (CDH) problem is hard. We will use an additive notation to describe the operation in $\mathcal{G}$ while we will use a multiplicative notation for the operation in $\mathcal{F}$. In practice, the group $\mathcal{G}$ is implemented using a group of points on certain

elliptic curves, each of which has a small MOV exponent [81] while the group $\mathcal{F}$ will be implemented using a subgroup of the multiplicative group of a finite field, that is, using the Weil pairing (whose elementary property is described in [21]) or the Tate pairing [53]. The admissible bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$ has the following properties.

- Bilinear: $\hat{e}(aR_1, bR_2) = \hat{e}(R_1, R_2)^{ab}$, where $R_1, R_2 \in \mathcal{G}$ and $a, b \in \mathbb{Z}_q^*$.
- Non-degenerate: $\hat{e}$ does not send all pairs of points in $\mathcal{G} \times \mathcal{G}$ to the identity in $\mathcal{F}$. (Hence, if R is a generator of $\mathcal{G}$ then $\hat{e}(R, R)$ is a generator of $\mathcal{F}$.)
- Computable: For all $R_1, R_2 \in \mathcal{G}$, the map $\hat{e}(R_1, R_2)$ is efficiently computable.

Throughout the rest of this thesis, we will simply use the term "Bilinear map" to refer to the admissible bilinear map defined above. We remark that the recent results [13, 57] show that the Bilinear map implemented using the Tate pairing offers better performance than the one implemented using the Weil pairing.

5.2.2 Boneh and Franklin's ID-Based Encryption Scheme

We now describe Boneh and Franklin's basic version of ID-based encryption scheme called "BasicIdent" which only gives semantic security (equivalently known as indistinguishability under chosen-plaintext attack). Below, we denote $\mathcal{G} \setminus \{O\}$ where O is the identity element of $\mathcal{G}$, and $\mathbb{Z}_q \setminus \{0\}$ by $\mathcal{G}^*$ and $\mathbb{Z}_q^*$ respectively. Also, we denote a security parameter by $k \in \mathbb{N}$.

- A randomized key/common parameter generation algorithm $\mathsf{GC}(k)$: This algorithm is run by the PKG to generate its master/public key pair and all the necessary common parameters.
  - Choose a group $\mathcal{G}$ of prime order q, whose generator is P. Specify the Bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$.
  - Pick a master key s uniformly at random from $\mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
  - Choose two hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathcal{G}^*$ and $\mathsf{H}_2 : \mathcal{F} \rightarrow \{0,1\}^l$, where l denotes the length of a plaintext.
  - Keep s as secret and publish a common parameter $cp = (\mathcal{G}, q, P, P_{pub}, \hat{e}, \mathsf{H}_1, \mathsf{H}_2)$.

- A deterministic private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: This algorithm is run by the PKG on receiving a private key extraction query from any user who wants to extract a private key that matches to an identity $\mathtt{ID}$.
  - Given $\mathtt{ID}$, compute $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = sQ_{\mathtt{ID}}$.
  - Output $D_{\mathtt{ID}}$.
- A randomized encryption algorithm $\mathsf{E}(cp, \mathtt{ID}, M)$: This algorithm is run by any user who wants to encrypt a message M using the receiver's identity $\mathtt{ID}$.
  - Compute $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$.
  - Pick r uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $U = rP$ and $V = \mathsf{H}_2(\hat{e}(Q_{\mathtt{ID}}, Y_{\mathrm{PKG}})^r) \oplus M$.
  - Return a ciphertext $C = (U, V)$.
- A deterministic decryption algorithm $\mathsf{D}(cp, D_{\mathtt{ID}}, C)$: This algorithm is run by the user who possesses a private key associated with an identity $\mathtt{ID}$, to decrypt C.
  - Parse C as (U, V).
  - Compute $M = V \oplus \mathsf{H}_2(\hat{e}(D_{\mathtt{ID}}, U))$.
  - Return M.

Note that in [27], Boneh and Franklin converted BasicIdent into "FullIdent" using the Fujisaki- Okamoto transform [55] to give chosen ciphertext security.

5.3 Related Work and Discussions

5.3.1 Threshold Decryption in the Non-ID-Based Setting

Earlier Constructions. The first construction of a threshold decryption scheme in the non-ID-based setting dates back to 1989. In [44], Desmedt and Frankel proposed a threshold decryption scheme based on the ElGamal encryption scheme [49]. Later, De Santis et al. [43] proposed a scheme based on the RSA problem [103]. However, the problem of those schemes is that they seem to be secure

against chosen-plaintext attack but not known to be secure against chosen ciphertext attack.

Chosen-Ciphertext Security for Threshold Decryption. In 1993, Lim and Lee [77] made an important observation on the security of threshold decryption that it is difficult to build a threshold decryption scheme withstanding chosen ciphertext attack without making it *publicly checkable*.

However, it was Shoup and Gennaro [111]'s work that first formalized a chosen ciphertext security notion for threshold decryption schemes and constructed two provably secure yet practical threshold decryption schemes. According to [112], they had tried to build up a publicly checkable threshold decryption scheme, which is necessary to construct a threshold decryption scheme secure against chosen ciphertext attack, by making the IND-CPA secure ElGamal-type threshold scheme publicly checkable using a non-interactive zero-knowledge proof of knowledge of discrete-logarithm which is similar to the Schnorr signature [104]. In fact, this method had been used by Tsiounis and Yung to design their public key encryption scheme [121]. But Shoup and Gennaro soon realized that it is difficult to prove such publicly checkable schemes (including Tsiounis and Yung's) are secure against chosen ciphertext attack, even in the random oracle model, due to the exponential blowing up of running time of rewinding the attacker to extract a knowledge. (For a more detailed account on this, readers are referred to [112].) They finally used a non-interactive zero-knowledge proof of membership to fix the problem, but this made their schemes somewhat less efficient.

After Shoup and Gennaro's work, Canetti and Goldwasser [29] proposed a threshold decryption scheme derived from Cramer and Shoup [41]'s famous public key encryption scheme provably secure against chosen ciphertext attack without depending on the random oracle model. Even though Cramer and Shoup's scheme is not publicly checkable, Canetti and Goldwasser used the algebraic property of the scheme that the receiver can check the validity of a ciphertext by using one part of the private key, before decrypting the ciphertext using the second part of the private key. However, the problem of their approach is that the servers must keep a large number of pre-shared secrets.

More recently, Fouque and Pointcheval [52] proposed a generic method to convert any public key encryption which is indistinguishable [62] against chosen-plaintext attack (IND-CPA) to the threshold decryption scheme secure against chosen ciphertext attack in the random oracle model. They observed that Naor and Yung [85]'s twin-encryption technique and the random oracle can yield an efficient non-interactive zero-knowledge proof of membership and as a result, threshold decryption schemes can be constructed. The main advantage of Fouque and

Pointcheval's conversion method is its generic nature that can be applied to various computational primitives such as integer factorization, but it has a drawback in that the converted threshold decryption schemes are complex and suffer a significant increase of the size of ciphertexts due to the use of the twin-encryption method.

5.3.2 Threshold Decryption in the ID-Based Setting

Boneh and Franklin's "Distributed PKG". In order to prevent a single PKG from full possession of the master key in ID-based encryption, Boneh and Franklin [27] suggested that the PKG's master key should be shared among a number of PKGs using the techniques of threshold cryptography, which they call "Distributed PKG". More precisely, the PKG's master key x is distributed into a number of PKGs in such a way that each of the PKG holds a share $x_i \in \mathbb{Z}_q^*$ of a Shamir's (t, n) secret-sharing [107] of $x \in \mathbb{Z}_q^*$ and responds to a user's private key extraction request with $D_{\mathtt{ID}}^i = x_i Q_{\mathtt{ID}}$, where $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$. If the technique of [60] is used, one can ensure that the master key is jointly generated by PKGs so that the master key is not stored or computed in any single location.

As an extension of the above technique, Boneh and Franklin suggested that the distributed PKGs should function as decryption servers for threshold decryption. That is, each PKG responds to a decryption query $C = (U, V)$ in BasicIdent with $\hat{e}(x_i Q_{\mathtt{ID}}, U)$. However, we argue that this method is not quite practical in practice since it requires each PKG to be involved *at all times* (that is, *on-line*) in the generation of decryption shares because the value "U" changes whenever a new ciphertext is created. Obviously, this creates a bottleneck on the PKGs and also violates one of the basic requirements of an ID-based encryption scheme, "the PKG can be closed after key generation", which was envisioned by Shamir in his original proposal of ID-based cryptography [108]. Moreover, there is a scalability problem when the number of available distributed PKGs is not matched against the number of decryption servers required, say, there are only 3 available PKGs while a certain application requires 5 decryption servers.

Therefore, a better approach would be *sharing a private key associated with an identity* rather than sharing a master key of the PKG. In addition to its easy adaptability to the situation where an identity denotes a group sharing a decryption key as described in Section 5.1, an advantage of this approach is that one can fully utilize Boneh and Franklin's Distributed PKG method without the above-mentioned scalability problem, dividing the role of "distributed PKGs" from that of "decryption servers". That is, an authorized dealer (a representative of group,

such as "Bob" described in Section 5.1, or a single PKG) may ask an identity to each of the "distributed PKGs" for a *partial* private key associated the identity. Having obtained enough partial private keys, the dealer can construct the whole private key and distribute it into the "decryption servers" in his domain at will while the master key remains secret from any parties.

Dodis and Yung, and Libert and Quisquiter's Work. To our knowledge, other papers that have treated "threshold decryption" in the context of ID-based cryptography are [47] and [75].

Dodis and Yung observed in [47] how threshold decryption can be realized in Gentry and Silverberg [61]'s "hierarchical ID-based encryption" setting. Interestingly, their approach is to share a private key (not the master key of the PKG) obtained from a user at a higher level. Although this was inevitable in the hierarchical ID-based encryption setting and its advantage in general ID-based cryptography was not mentioned in [47], it is more sound approach than sharing the master key of the PKG as we discussed above. However, their threshold decryption scheme is very-sketched and chosen ciphertext security for the scheme was not considered in [47].

More recently, Libert and Quisquater [75] also constructed an ID-based threshold decryption scheme. However, their approach was to share a master key of the PKG, which is different from ours. Moreover, our scheme gives chosen ciphertext security while Libert and Quisquater's scheme does not.

Other Related Work. Although not being directly related to "ID-based threshold decryption", there have been interesting applications of Boneh and Franklin's Distributed PKG.

Recently, Chen, Harrison, Soldera, and Smart [38] illustrated how the distributed PKGs in Boneh and Franklin's ID-based encryption can be applied to the real world situations. Furthermore, they dealt with general cases of disjunction and conjunction of the multiple PKGs/identities exploiting, the algebra of the Bilinear maps. Subsequently, more complicated cases of the disjunction and conjunction of the multiple PKGs and their applications to access controls were discussed by Smart [113].

Khalili, Katz and Arbaugh [71] also discussed the use of the distributed PKGs in Boneh and Franklin's scheme, especially focusing on its application to ad-hoc networks.

5.4 Security Notion for ID-based Threshold Decryption

5.4.1 High Level Description of ID-based Threshold Decryption

We denote a generic (t, n) ID-based threshold decryption scheme by "$\mathcal{IDTHD}$", which consists of algorithms GC, EX, DK, E, D, SV, and SC.

Like other ID-based cryptographic schemes, we assume the existence of a trusted PKG. The PKG runs the key/common parameter generation algorithm GC to generate its master/public key pair and all the necessary common parameters. The PKG's public key and the common parameters are given to every interested party.

On receiving a user's private key extraction request which consists of an identity, the PKG then runs the private key extraction algorithm EX to generate the private key associated with the requested identity.

An authorized dealer who possesses the private key associated with an identity can run the private key distribution algorithm DK to distribute the private key into n decryption servers. DK makes use of an appropriate secret-sharing technique to generate shares of the private key as well as verification keys that will be used for checking the validity of decryption shares. Each share of the private key and its corresponding verification key are sent to an appropriate decryption server. The decryption servers then keep their private key shares secret but publish the verification keys. It is important to note here that the entity that runs DK can vary flexibly depending on the cryptographic services that the PKG can offer. For example, if the PKG has an only functionality of issuing private keys, the authorized dealer that runs DK would be a normal user (such as Bob in the example given in Section 5.1) other than the PKG. However, if the PKG has other functionalities, for example, organizing threshold decryption, the PKG can run DK.

Given a user's identity, any user that wants to encrypt a plaintext can run the encryption algorithm E to obtain a ciphertext. A *legitimate* user that wants to decrypt a ciphertext gives it to the decryption servers requesting decryption shares. The decryption servers then run the decryption share generation algorithm D taking the ciphertext as input and send the resulting decryption shares to the user. Note that the validity of the shares can be checked by running the

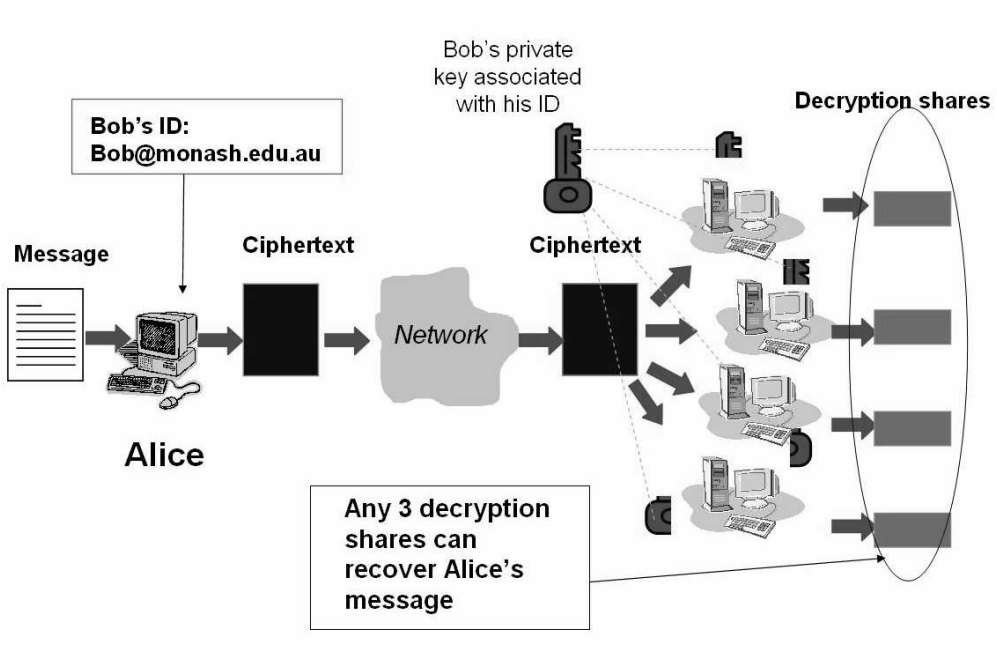


Figure 5.1: (3,4)-Identity-Based Threshold Decryption

decryption share verification algorithm SV. When the user collects valid decryption shares from at least t servers, the plaintext can be reconstructed by running the share combining algorithm SC.

Figure 5.1 illustrates the scenario of $(3, 4)$-ID-based threshold decryption. Below, we formally define $\mathcal{IDTHD}$.

Definition 15 (ID-Based Threshold Decryption) The $\mathcal{IDTHD}$ scheme consists of the following algorithms.

- A randomized key/common parameter generation algorithm $\mathsf{GC}(k)$: Given a security parameter $k \in \mathbb{N}$, this algorithm computes the PKG's master/public key pair $(sk_{\mathrm{PKG}}, pk_{\mathrm{PKG}})$. Then, it generates necessary common parameters, e.g., descriptions of hash functions and mathematical groups. The output of this algorithm denoted by cp includes such parameters and the PKG's public key pk_{PKG}. Note that cp is given to all interested entities while the matching master key sk_{PKG} of pk_{PKG} is kept secret.
- A private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: Given an identity $\mathtt{ID}$, this algorithm generates a private key associated with $\mathtt{ID}$, denoted by $sk_{\mathtt{ID}}$.

- A randomized private key distribution algorithm $\mathsf{DK}(cp, sk_{\mathtt{ID}}, n, t)$: Given a private key $sk_{\mathtt{ID}}$ associated with an identity $\mathtt{ID}$, a number of decryption servers n and a threshold parameter t, this algorithm generates n shares of $sk_{\mathtt{ID}}$ and provides each one to decryption servers $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$. It also generates a set of verification keys that can be used to check the validity of each shared private key. We denote the shared private keys and the matching verification keys by $\{sk_i\}_{1 \leq i \leq n}$ and $\{vk_i\}_{1 \leq i \leq n}$, respectively. Note that for each $1 \leq i \leq n$, the pair (sk_i, vk_i) is sent to the decryption server Γ_i, then Γ_i publishes vk_i but keeps sk_i as secret.

- A randomized encryption algorithm $\mathsf{E}(cp, \mathtt{ID}, M)$: Given a public identity $\mathtt{ID}$ and a plaintext M, this algorithm generates a ciphertext denoted by C.

- A decryption share generation algorithm $\mathsf{D}(cp, sk_i, C)$: Given a ciphertext C and a shared private key sk_i of a decryption server Γ_i, this algorithm generates a decryption share $\delta_{i,C}$. Note that the value of $\delta_{i,C}$ can be a special symbol "*Invalid Ciphertext*".

- A decryption share verification algorithm $\mathsf{SV}(cp, \{vk_i\}_{1 \leq i \leq n}, C, \delta_{i,C})$: Given a ciphertext C, a set of verification keys $\{vk_i\}_{1 \leq i \leq n}$, and a decryption share $\delta_{i,C}$, this algorithm checks the validity of $\delta_{i,C}$. The output of this algorithm is either "*Valid Share*" or "*Invalid Share*".

- A share combining algorithm $\mathsf{SC}(cp, C, \{\delta_{i,C}\}_{i \in \Phi})$: Given a ciphertext C and a set of decryption shares $\{\delta_{i,C}\}$ where $\Phi \subset \{1, \ldots, n\}$ such that $|\Phi| \geq t$ ($|\cdot|$ denotes the cardinality), this algorithm outputs a plaintext M. Note that the combining algorithm is allowed to output a special symbol "*Invalid Ciphertext*", which is distinct from all possible plaintexts.

5.4.2 Chosen-Ciphertext Security for ID-Based Threshold Decryption

We now define a security notion for the $\mathcal{IDTHD}$ scheme against chosen ciphertext attack, which we call "IND-IDTHD-CCA".

Definition 16 (IND-IDTHD-CCA) Let $\mathsf{A}^{\mathsf{CCA}}$ be an attacker assumed to be a probabilistic Turing machine. Suppose that a security parameter k is given to $\mathsf{A}^{\mathsf{CCA}}$ as input. Now, consider the following game in which the attacker $\mathsf{A}^{\mathsf{CCA}}$ interacts with the "Challenger".

Phase 1: The Challenger runs the PKG's key/common parameter generation algorithm taking a security parameter k as input. The Challenger gives $\mathsf{A}^{\mathsf{CCA}}$ the resulting common parameter cp which includes the PKG's public key pk_{PKG}. However, the Challenger keeps the master key sk_{PKG} secret from $\mathsf{A}^{\mathsf{CCA}}$.

Phase 2: $\mathsf{A}^{\mathsf{CCA}}$ issues a number of private key extraction queries. We denote each of these queries by $\mathtt{ID}$. On receiving the identity query $\mathtt{ID}$, the Challenger runs the private key extraction algorithm on input $\mathtt{ID}$ and obtains a corresponding private key $sk_{\mathtt{ID}}$. Then, the Challenger returns $sk_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 3: $\mathsf{A}^{\mathsf{CCA}}$ corrupts $t-1$ out of n decryption servers.

Phase 4: $\mathsf{A}^{\mathsf{CCA}}$ issues a target identity query $\mathtt{ID}^*$. On receiving $\mathtt{ID}^*$, the Challenger runs the private key extraction algorithm to obtain a private key $sk_{\mathtt{ID}^*}$ associated with the target identity. The Challenger then runs the private key distribution algorithm on input $sk_{\mathtt{ID}^*}$ with parameter (t, n) and obtains a set of private/verification key pairs $\{(sk_{\mathtt{ID}^*{}_i}, vk_{\mathtt{ID}^*{}_i})\}$, where $1 \leq i \leq n$. Next, the Challenger gives $\mathsf{A}^{\mathsf{CCA}}$ the private keys of corrupted decryption servers and the verifications keys of all the decryption servers. However, the private keys of uncorrupted servers are kept secret from $\mathsf{A}^{\mathsf{CCA}}$.

Phase 5: $\mathsf{A}^{\mathsf{CCA}}$ issues arbitrary private key extraction queries and arbitrary decryption share generation queries to the uncorrupted decryption servers. We denote each of these queries by $\mathtt{ID}$ and C respectively. On receiving $\mathtt{ID}$, the Challenger runs the private key extraction algorithm to obtain a private key associated with $\mathtt{ID}$ and returns it to $\mathsf{A}^{\mathsf{CCA}}$. The only restriction here is that $\mathsf{A}^{\mathsf{CCA}}$ is not allowed to query the target identity $\mathtt{ID}^*$ to the private key extraction algorithm. On receiving C, the Challenger runs the decryption share generation algorithm taking C and the target identity $\mathtt{ID}^*$ as input to obtain a corresponding decryption share and returns it to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 6: $\mathsf{A}^{\mathsf{CCA}}$ outputs two equal-length plaintexts (M_0, M_1). Then the Challenger chooses a bit β uniformly at random and runs the encryption algorithm on input cp, M_β and $\mathtt{ID}^*$ to obtain a target ciphertext $C^* = \mathsf{E}(cp, \mathtt{ID}^*, M_\beta)$. Finally, the Challenger gives $(C^*, \mathtt{ID}^*)$ to $\mathsf{A}^{\mathsf{CCA}}$.

Phase 7: $\mathsf{A}^{\mathsf{CCA}}$ issues arbitrary private key extraction queries and arbitrary decryption share generation queries. We denote each of these queries by $\mathtt{ID}$ and C respectively. On receiving $\mathtt{ID}$, the Challenger runs the private key extraction algorithm to obtain a private key associated with $\mathtt{ID}$ and returns

it to $\mathsf{A}^{\mathsf{CCA}}$. As Phase 5, the only restriction here is that $\mathsf{A}^{\mathsf{CCA}}$ is not allowed to query the target identity $\mathtt{ID}^*$ to the private key extraction algorithm. On receiving C, the Challenger runs the decryption share generation algorithm on input C to obtain a corresponding decryption share and returns it to $\mathsf{A}^{\mathsf{CCA}}$. Differently from Phase 5, the target ciphertext C^* is not allowed to query in this phase.

Phase 8: $\mathsf{A}^{\mathsf{CCA}}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$.

We define the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success by

$$\mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathcal{IDTHD},\mathsf{A}^{\mathsf{CCA}}}(k) = 2 \cdot \Pr[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathcal{IDTHD}}(t_{IDCCA}, q_E, q_D)$ the maximum of the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CCA}}$ having running time t_{IDCCA} and making at most q_E private key extraction queries and q_D decryption share generation queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The ID-based threshold decryption scheme $\mathcal{IDTHD}$ is said to be IND-IDTHD-CCA secure if $\mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathcal{IDTHD}}(t_{IDCCA}, q_E, q_D)$ is negligible in k.

5.5 Our ID-Based Threshold Decryption Scheme

5.5.1 Building Blocks

First, we present necessary building blocks that will be used to construct our ID-based threshold decryption scheme. We remark that since our ID-based threshold decryption scheme is also of the Diffie-Hellman (DH)-type, it follows Shoup and Gennaro [112]'s framework for the design of DH-based threshold decryption schemes to some extent. However, our scheme has a number of features that distinguishes itself from the schemes in [112] due to the special property of the underlying group $\mathcal{G}$.

Publicly Checkable Encryption

Publicly checkable encryption is a particularly important tool for building threshold decryption schemes secure against chosen ciphertext attack as discussed by

Lim and Lee [77]. The main reason is that in the threshold decryption, the attacker has decryption shares as additional information as well as a ciphertext, hence there is a big chance for the attacker to get enough decryption shares to recover the plaintext before the validity of the ciphertext is checked. (Readers are referred to [77] and [112] for more detailed discussions on this issue.)

Note that public checkability of ciphertexts in threshold decryption schemes is usually given by non-interactive zero-knowledge (NIZK) proofs, e.g., [112, 52]. However, we emphasize that in our scheme, this can be done *without* a NIZK proof, by simply creating a tag on the ElGamal [49] ciphertext as follows.

Let $M \in \{0,1\}^l$ be a message. Then, encrypt M by creating a ciphertext $C = (U, V, W) = (rP, \mathsf{H}_2(\kappa) \oplus M, r\mathsf{H}_3(U, V))$ where $\kappa = \hat{e}(\mathsf{H}_1(\mathtt{ID}), Y_{\mathrm{PKG}})^r$ for hash functions $\mathsf{H}_1 : \{0,1\}^* \to \mathcal{G}^*$, $\mathsf{H}_2 : \mathcal{F} \to \{0,1\}^l$, and $\mathsf{H}_3 : \mathcal{G}^* \times \{0,1\}^l \to \mathcal{G}^*$. Without recovering M during the decryption process (that is, leaving the ciphertext C intact), the validity of C can be checked by testing if $\hat{e}(P, W) = \hat{e}(U, H_3)$, where $H_3 = \mathsf{H}_3(U, V) \in \mathcal{G}^*$. Note that this validity test exploits the fact that the Decisional Diffie-Hellman (DDH) problem can be solved in polynomial time in the group $\mathcal{G}$, and passing the test implies that (P, U, H_3, W) is a Diffie-Hellman tuple since $(P, U, H_3, W) = (P, rP, sP, rsP)$ assuming that $H_3 = sP \in_R \mathcal{G}^*$ for some $s \in \mathbb{Z}_q^*$.

We remark that using the above property of the Bilinear map, we could develop the non-ID-based threshold decryption scheme which is simple and provides short ciphertexts [9].

Secret-Sharing over $\mathcal{G}$

Recall that the distributed PKGs of Boneh and Franklin's ID-based encryption scheme can be achieved by sharing the PKG's master key x. Indeed, this can be done using Shamir's secret-sharing technique [107] as the master key x is a single element in $\mathbb{Z}_q^*$ and hence Shamir's technique can directly be used to distribute an element in $\mathbb{Z}_q^*$.

However, in order to distribute a private key $D_{\mathtt{ID}} \in \mathcal{G}$, we need some trick. In what follows, we show one can easily share a point on $\mathcal{G}$ by modifying Shamir's (t, n) secret-sharing scheme. (Note that although "Shamir's secret-sharing over $\mathcal{G}$" was mentioned in [47], how to realize it was not described. So we explicitly describe it for clarity.)

Distribution Phase: Let q be a prime order of a group $\mathcal{G}$ of points on elliptic curve. Let $S \in \mathcal{G}^*$ be a secret-point to share. Suppose that we have chosen integers t and n satisfying $1 \leq t \leq n < q$.

First, we pick $R_1, R_2, \ldots, R_{t-1}$ at random from $\mathcal{G}^*$. Then, we define a function $F : \mathbb{N} \cup \{0\} \to \mathcal{G}$ such that $F(u) = S + \sum_{l=1}^{t-1} u^l R_l$.

Now, we compute $S_i = F(i) \in \mathcal{G}$ for $1 \leq i \leq n$ and send (i, S_i) to the i-th member of the group of cardinality n. Note that when $i = 0$, we obtain the secret itself, that is, $S = F(0)$. (We assume that the "multiplication-by-m map for a positive integer m" denoted by mP is extended to all integer $m \in \mathbb{Z}$ by defining $0P = O$ where O is the identity element of $\mathcal{G}$, and $(-m)P = -(mP)$ [21].) Note that in practice, "picking R_l at random from $\mathcal{G}^*$" can be implemented by computing $r_l P$ for randomly chosen $r_l \in \mathbb{Z}_q^*$, where $P \in \mathcal{G}^*$ is a generator of $\mathcal{G}$.

Reconstruction Phase: Let $\Phi \subset \{1, \ldots, n\}$ be a set such that $|\Phi| \geq t$, where $|\cdot|$ denotes the cardinality of the given set. The function $F(u)$ can be reconstructed by computing

$$F(u) = \sum_{j \in \Phi} c_{uj}^{\Phi} S_j \text{ where } c_{uj}^{\Phi} = \prod_{\iota \in \Phi, \iota \neq j} \frac{u - \iota}{j - \iota} \in \mathbb{Z}_q.$$

Notice that $c_{uj}^{\Phi} \in \mathbb{Z}_q$ is the Lagrange interpolation coefficient used in Shamir's secret sharing scheme: If we write $S = sP$ and $R_l = r_l P$ for for some $s, r_l \in \mathbb{Z}_q^*$ and $1 \leq l \leq t-1$ (but, we do not know s and r_l), we have $F(u) = sP + ur_1 P + \cdots + u^{t-1} r_{t-1} P = (s + r_1 u + \cdots + r_{t-1} u^{t-1})P$. Hence, the Lagrange coefficients c_{uj}^{Φ}'s reconstruct the original function $F(u)$. In practice, we recover the secret S directly (without reconstructing $F(u)$) by computing $\sum_{j \in \Phi} c_{0j}^{\Phi} S_j$ where $c_{0j}^{\Phi} = \prod_{\iota \in \Phi, \iota \neq j} \frac{\iota}{\iota - j}$. Note that the computation of $c_{ij}^{\Phi} \in \mathbb{Z}_q$ can be done in polynomial time.

Zero Knowledge Proof for the Equality of Two Discrete-Logarithms Based on the Bilinear Map

To ensure that all decryption shares are consistent, that is, to give robustness to threshold decryption, we need a certain checking procedure. In contrast to the validity checking method of ciphertexts discussed in Section 5.5.1, we need a non-interactive zero-knowledge proof system since the share of the key κ is the element of the group $\mathcal{F}$, where the DDH problem is believed to be hard.

Motivated by [37] and [112], we construct a zero-knowledge proof of membership system for the language $L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}} \stackrel{\text{def}}{=} \{(\mu, \tilde{\mu}) \in \mathcal{F} \times \mathcal{F} |\ \log_g \mu = \log_{\tilde{g}} \tilde{\mu}\}$ where $g = \hat{e}(P, P)$ and $\tilde{g} = \hat{e}(P, \tilde{P})$ for generators P and $\tilde{P}$ of $\mathcal{G}$ (the groups $\mathcal{G}$ and $\mathcal{F}$, and the Bilinear map $\hat{e}$ are as defined in Section 5.2.2) as follows.

Suppose that $(P, \tilde{P}, g, \tilde{g})$ and $(\kappa, \tilde{\kappa}) \in L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$ are given to the Prover and the Verifier, and the Prover knows a secret $S \in \mathcal{G}^*$. The proof system which we call "ZKBM" works as follows.

- The Prover chooses a non-identity element T uniformly at random from $\mathcal{G}$ and computes $\gamma = \hat{e}(T, P)$ and $\tilde{\gamma} = \hat{e}(T, \tilde{P})$. The Prover sends γ and $\tilde{\gamma}$ to the Verifier.
- The Verifier chooses h uniformly at random from $\mathbb{Z}_q^*$ and sends it to the Prover.
- On receiving h, the Prover computes $L = T + hS \in \mathcal{G}$ and sends it to the Verifier. The Verifier checks if $\hat{e}(L, P) = \gamma\kappa^h$ and $\hat{e}(L, \tilde{P}) = \tilde{\gamma}\tilde{\kappa}^h$. If the equality holds then the Verifier returns "*Accept*", otherwise, returns "*Reject*".

We state the following lemma regarding the security of ZKBM.

Lemma 6 *The* ZKBM *protocol satisfies completeness, soundness and zero-knowledge against the honest Verifier.*

Proof. As preliminaries, we first prove the following two claims.

Claim 1 *Let P and $\tilde{P}$ be generators of $\mathcal{G}$. Then $\hat{e}(P, \tilde{P})$ is a generator of $\mathcal{F}$.*

Proof. The proof will use the basic fact from the elementary abstract algebra that if a is a generator of a finite cyclic group G of order n, then the other generators of G are the elements of the form a^r, where $gcd(r, n) = 1$.

First, note that the two groups $\mathcal{G}$ and $\mathcal{F}$ are cyclic because their order q is a prime. Since $\tilde{P}$ is another generator of $\mathcal{G}$ by assumption, we can write $\tilde{P} = uP$, where $gcd(u, q) = 1$. Then, by the bilinear property of $\hat{e}$, we have $\hat{e}(P, \tilde{P}) = \hat{e}(P, uP) = \hat{e}(P, P)^u$. Also, by the non-degenerate property of $\hat{e}$, $\hat{e}(P, P)$ is a generator of $\mathcal{F}$. Hence, $\hat{e}(P, \tilde{P})$ is also a generator of $\mathcal{F}$ since $\hat{e}(P, \tilde{P}) = \hat{e}(P, P)^u$ and $gcd(u, q)$=1. □

Claim 2 *Let P and $\tilde{P}$ be generators of $\mathcal{G}$. Then, $(\kappa, \tilde{\kappa}) \in L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$ if and only if there exists a non-identity element $S \in \mathcal{G}$ such that $\kappa = \hat{e}(S, P)$ and $\tilde{\kappa} = \hat{e}(S, \tilde{P})$.*

Proof. By Claim 1, g and $\tilde{g}$ are generators of $\mathcal{F}$. Now, suppose that $(\kappa, \tilde{\kappa}) \in L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$. Then, by definition of $L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$, there exists $x \in \mathbb{Z}_q^*$ such that $g^x = \tilde{g}^x$. Since $g = \hat{e}(P, P)$ and $\tilde{g} = \hat{e}(P, \tilde{P})$, $g^x = \tilde{g}^x$ implies $\hat{e}(P, P)^x = \hat{e}(P, \tilde{P})^x$. But, since $\hat{e}(P, P)^x = \hat{e}(xP, P)$ and $\hat{e}(P, \tilde{P})^x = \hat{e}(xP, \tilde{P})$ by the bilinear property of $\hat{e}$, we obtain $\kappa = \hat{e}(S, P)$ and $\tilde{\kappa} = \hat{e}(S, \tilde{P})$ by letting $S = xP$. The proof of converse is also easy. □

Now, we show that the protocol is complete. That is, if the Prover and the Verifier follow the protocol without cheating, the Verifier accepts the Prover's claim with overwhelming probability: Assume that $(\kappa, \tilde{\kappa}) \in L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$. By Claim 2, we have $\kappa = \hat{e}(S, P)$ and $\tilde{\kappa} = \hat{e}(S, \tilde{P})$ for some $S \in \mathcal{G}$. Assume that the Prover sends $(\gamma, \tilde{\gamma})$ where $\gamma = \hat{e}(T, P)$ and $\tilde{\gamma} = \hat{e}(T, \tilde{P})$ for random $T \in \mathcal{G}$ to the honest Verifier. Now, observe from the above protocol that $\hat{e}(L, P) = \hat{e}(T + hS, P)$ and that $\gamma\kappa^h = \hat{e}(T, P)\hat{e}(S, P)^h = \hat{e}(T, P)\hat{e}(hS, P)$. By the bilinear property of $\hat{e}$, we have $\hat{e}(T, P)\hat{e}(hS, P) = \hat{e}(T + hS, P)$. Thus, we obtain $\hat{e}(L, P) = \gamma\kappa^h$ and this implies that the above protocol satisfies completeness property.

Second, we show the soundness of the protocol: Assume that $(\kappa, \tilde{\kappa}) \notin L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$. Namely, we have $\kappa = \hat{e}(S, P)$ and $\tilde{\kappa} = \hat{e}(S', \tilde{P})$ for some $S \neq S' \in \mathcal{G}$. Assume that a cheating Prover sends $(\gamma, \tilde{\gamma})$ where $\gamma = \hat{e}(T, P)$ and $\gamma = \hat{e}(T', \tilde{P})$ to the honest Verifier. If the Verifier is to accept this, we should have that $\hat{e}(L, P) = \gamma\kappa^h$ and $\hat{e}(L, \tilde{P}) = \tilde{\gamma}\tilde{\kappa}^h$, which implies $T + hS = T' + hS'$. Now suppose that $T = tP$, $T' = t'P$; $S = xP$ and $S' = x'P$ for $t, t', x, x' \in \mathbb{Z}_q^*$. Then, $T + hS = T' + hS'$ implies $(t - t') + h(x - x') = 0$. However, this happens with probability $1/q$, since we have assumed that $S' \neq S$ which implies $x' \neq x$.

Finally, we can construct a simulator which simulates the communication between the Prover and the Verifier provided that the Verifier behaves *honestly*. More precisely, the simulator chooses $\bar{h}$ and $\bar{L}$ uniformly at random from $\mathbb{Z}_q^*$ and $\mathcal{G}$ respectively. Then, it computes $\bar{\gamma} = \hat{e}(\bar{L}, P)/\kappa^{\bar{h}}$ and $\bar{\tilde{\gamma}} = \hat{e}(\bar{L}, \tilde{P})/\tilde{\kappa}^{\bar{h}}$. The output of the simulator is a tuple $(\bar{\gamma}, \bar{\tilde{\gamma}}, \bar{h}, \bar{L})$. It can be easily verified that the simulated values are identically distributed as those in the real communication if the Verifier behaves honestly. As a result, the above protocol becomes a zero-knowledge proof against a honest Verifier. □

Notice that ZKBM can easily be converted to a NIZK proof, making the random challenge an output of a random oracle [20]. Note that the above protocol can be viewed as a proof that $(g, \tilde{g}, \kappa, \tilde{\kappa})$ is a Diffie-Hellman tuple since if $(\kappa, \tilde{\kappa}) \in L_{\mathsf{EDLog}^{\mathcal{F}}_{P,\tilde{P}}}$ then $\kappa = g^x$ and $\tilde{\kappa} = \tilde{g}^x$ for some $x \in \mathbb{Z}_q^*$ and hence $(g, \tilde{g}, \kappa, \tilde{\kappa}) = (g, \tilde{g}, g^x, \tilde{g}^x) = (g, g^y, g^x, g^{xy})$ for some $y \in \mathbb{Z}_q^*$.

5.5.2 Description of Our ID-Based Threshold Decryption Scheme

We now describe our ID-based threshold decryption scheme. We call our scheme "IDTHDBM", meaning "ID-based threshold decryption scheme from the bilinear map". The IDTHDBM scheme consists of the following algorithms.

- $\mathsf{GC}(k)$: Given a security parameter k, this algorithm generates two groups $\mathcal{G}$ and $\mathcal{F}$ of the same prime order $q \geq 2^k$ and chooses a generator P of $\mathcal{G}$. Then, it specifies the Bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$ and the hash functions H_1, H_2, H_3 and H_4 such that $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathcal{G}^*$; $\mathsf{H}_2 : \mathcal{F} \rightarrow \{0,1\}^l$; $\mathsf{H}_3 : \mathcal{G}^* \times \{0,1\}^l \rightarrow \mathcal{G}^*$; $\mathsf{H}_4 : \mathcal{F} \times \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{Z}_q^*$, where l denotes the length of a plaintext. Next, it chooses the PKG's master key x uniformly at random from $\mathbb{Z}_q^*$ and computes the PKG's public key $Y_{\mathrm{PKG}} = xP$. Finally, it returns a common parameter $cp = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y_{\mathrm{PKG}})$ while keeping the master key x secret.

- $\mathsf{EX}(cp, \mathtt{ID})$: Given an identity ID, this algorithm computes $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = xQ_{\mathtt{ID}}$. Then, it returns the private key $D_{\mathtt{ID}}$ associated with ID.

- $\mathsf{DK}(cp, \mathtt{ID}, D_{\mathtt{ID}}, t, n)$ where $1 \leq t \leq n < q$: Given a private key $D_{\mathtt{ID}}$, the number of decryption servers n and a threshold parameter t, this algorithm first picks $R_1, R_2, \ldots, R_{t-1}$ at random from $\mathcal{G}^*$ and constructs $F(u) = D_{\mathtt{ID}} + \sum_{j=1}^{t-1} u^j R_j$ for $u \in \{0\} \cup \mathbb{N}$. It then computes each server Γ_i's private key $S_i = F(i)$ and verification key $y_i = \hat{e}(S_i, P)$ for $1 \leq i \leq n$. Subsequently, it secretly sends the distributed private key S_i and the verification key y_i to server Γ_i for $1 \leq i \leq n$. Γ_i then keeps S_i as secret while making y_i public.

- $\mathsf{E}(cp, \mathtt{ID}, m)$: Given a plaintext $M \in \{0,1\}^l$ and an identity ID, this algorithm chooses r uniformly at random from $\mathbb{Z}_q^*$, and subsequently computes $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $\kappa = \hat{e}(Q_{\mathtt{ID}}, Y_{\mathrm{PKG}})^r$. It then computes

$$U = rP; V = \mathsf{H}_2(\kappa) \oplus M; W = r\mathsf{H}_3(U, V)$$

and returns a ciphertext $C = (U, V, W)$.

- $\mathsf{D}(cp, S_i, C)$: Given a private key S_i of each decryption server and a ciphertext $C = (U, V, W)$, this algorithm computes $H_3 = \mathsf{H}_3(U, V)$ and checks if $\hat{e}(P, W) = \hat{e}(U, H_3)$.

  If C has passed the above test, this algorithm computes $\kappa_i = \hat{e}(S_i, U)$, $\tilde{\kappa_i} = \hat{e}(T_i, U)$, $\tilde{y_i} = \hat{e}(T_i, P)$, $\lambda_i = \mathsf{H}_4(\kappa_i, \tilde{\kappa_i}, \tilde{y_i})$, and $L_i = T_i + \lambda_i S_i$ for random $T_i \in \mathcal{G}$, and outputs $\delta_{i,C} = (i, \kappa_i, \tilde{\kappa_i}, \tilde{y_i}, \lambda_i, L_i)$. Otherwise, it returns $\delta_{i,C} =$(i, "*Invalid Ciphertext*").

- $\mathsf{SV}(cp, \{y_i\}_{1 \le i \le n}, C, \delta_{i,C})$: Given a ciphertext $C = (U, V, W)$, a set of verification keys $\{y_1, \ldots, y_n\}$, and a decryption share $\delta_{i,C}$, this algorithm computes $H_3 = \mathsf{H}_3(U, V)$ and checks if $\hat{e}(P, W) = \hat{e}(U, H_3)$.

  If C has passed the above test then this algorithm does the following:

  - If $\delta_{i,C}$ is of the form (i, "*Invalid Ciphertext*") then return "*Invalid Share*".
  - Else parse $\delta_{i,C}$ as $(i, \kappa_i, \tilde{\kappa_i}, \tilde{y_i}, \lambda_i, L_i)$ and compute $\lambda_i' = \mathsf{H}_4(\kappa_i, \tilde{\kappa_i}, \tilde{y_i})$.
    - Check if $\lambda_i' = \lambda_i$, $\hat{e}(L_i, U)/\kappa_i^{\lambda_i'} = \tilde{\kappa_i}$ and $\hat{e}(L_i, P)/y_i^{\lambda_i'} = \tilde{y_i}$.
    - If the test above holds, return "*Valid Share*", else output "*Invalid Share*".

  Otherwise, does the following:

  - If $\delta_{i,C}$ is of the form (i, "*Invalid Ciphertext*"), return "*Valid Share*", else output "*Invalid Share*".

- $\mathsf{SC}(cp, C, \{\delta_{j,C}\}_{j \in \Phi})$: Given a ciphertext C and a set of valid decryption shares $\{\delta_{j,C}\}_{j \in \Phi}$ where $|\Phi| \ge t$, this algorithm computes $H_3 = \mathsf{H}_3(U, V)$ and checks if $\hat{e}(P, W) = \hat{e}(U, H_3)$.

  If C has not passed the above test, this algorithm returns "*Invalid Ciphertext*". (In this case, all the decryption shares are of the form (i, "*Invalid Ciphertext*").) Otherwise, it computes $\kappa = \prod_{j \in \Phi} \kappa_j^{c_{0j}^{\Phi}}$ and $M = \mathsf{H}_2(\kappa) \oplus V$, and returns M.

It is easy to see that if C is a valid ciphertext and $|\Phi| \ge t$ then $\mathsf{SC}(C, \{\delta_j\}_{j \in \Phi}) = m$: Indeed, if $C = (U, V, W)$ has passed all the validity checks above,

$$\prod_{j \in \Phi} \kappa_j^{c_{0j}^{\Phi}} = \prod_{i \in \Phi} \hat{e}(S_j, U)^{c_{0j}^{\Phi}} = \prod_{j \in \Phi} \hat{e}(S_j, rP)^{c_{0j}^{\Phi}} = \hat{e}(\sum_{j \in \Phi} c_{0j}^{\Phi} S_j, rP)$$

$$
\begin{aligned}
&= \hat{e}(D_{\mathtt{ID}}, P)^r = \hat{e}(xQ_{\mathtt{ID}}, P)^r = \hat{e}(Q_{\mathtt{ID}}, xP)^r \\
&= \hat{e}(Q_{\mathtt{ID}}, Y_{\mathrm{PKG}})^r = d^r = \kappa,
\end{aligned}
$$

where c_{0j}^{Φ} is the Lagrange coefficient defined in Section 5.5.1. Hence, $\mathsf{H}_2(\kappa) \oplus V = \mathsf{H}_2(\kappa) \oplus (\mathsf{H}_1(d^r) \oplus M) = M$.

5.5.3 Security Analysis of Our ID-Based Threshold Decryption Scheme

Bilinear Diffie-Hellman Problem

Before analyzing our scheme, we review the Bilinear Diffie-Hellman (BDH) problem, which is a new class of computational problem introduced by Boneh and Franklin [27].

Definition 17 (BDH) Let $\mathcal{G}$ and $\mathcal{F}$ be two groups of a prime order $q \geq 2^k$, where k is security parameter. Let $P \in \mathcal{G}^*$ be a generator of $\mathcal{G}$. Suppose that there exists a bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$. Let $\mathsf{A}^{\mathsf{BDH}}$ be an attacker modelled as a probabilistic Turing machine taking the security parameter k as input. Suppose that a, b, and c are uniformly chosen at random from $\mathbb{Z}_q^*$ and aP, bP, and cP are computed.

$\mathsf{A}^{\mathsf{BDH}}$ is to solve the following problem:

- Given $(\mathcal{G}, q, \hat{e}, P, aP, bP, cP)$, compute $\hat{e}(P, P)^{abc}$.

We define $\mathsf{A}^{\mathsf{BDH}}$'s success by

$$
\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{BDH}}}(k) = \Pr[\mathsf{A}^{\mathsf{BDH}} \text{ outputs } \hat{e}(P, P)^{abc}].
$$

We denote by $\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G}}(t_{\mathsf{BDH}})$ the maximal success probability $\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{BDH}}}(k)$ over all attackers having running time bounded by t_{BDH} which is polynomial in the security parameter k.

The BDH problem is said to be computationally intractable if $\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G}}(t_{BDH})$ is negligible in k.

Proof of Security

Regarding the security of the IDTHDBM scheme, we obtain the following theorem implying that the IDTHDBM scheme is IND-IDTHD-CCA secure in the random oracle model assuming that the BDH problem is computationally intractable.

Theorem 5 *Suppose that an IND-IDTHD-CCA attacker for the scheme* IDTHDBM *issues up to q_E private key extraction queries, q_D decryption share generation queries, q_{H_1}, q_{H_2}, q_{H_3}, and q_{H_4} queries to to the random oracles H_1, H_2, H_3, and H_4 respectively. Using this attacker as a subroutine, we can construct an attacker for solving the BDH problem in the group $\mathcal{G}$, whose running time is bounded by t_{BDH}. Concretely, we obtain the following advantage bound:*

$$\frac{1}{q_{\mathsf{H}_1}}\mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathtt{IDTHDBM}}(t_{IDCCA}, q_E, q_D, q_{\mathsf{H}_1}, q_{\mathsf{H}_2}, q_{\mathsf{H}_3}, q_{\mathsf{H}_4}) \leq 2\mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G}}(t_{\mathrm{BDH}}) + \frac{q_D + q_D q_{\mathsf{H}_4}}{2^{k-1}},$$

where $t_{\mathrm{BDH}} = t_{IDCCA} + \max(q_E, q_{\mathsf{H}_1})O(k^3) + q_{\mathsf{H}_1} + q_{\mathsf{H}_2}O(k^3) + q_{\mathsf{H}_4}q_D O(k^3)$ for a security parameter k.

To prove the above theorem, we derive a non-ID-based threshold decryption scheme called "THDBM" from the IDTHDBM scheme, which will be described shortly. We then show in Lemma 7 that the IND-THD-CCA security of the THDBM scheme, which will be defined after the description of THDBM, implies the IND-IDTHD-CCA security of the IDTHDBM scheme. Next, we show in Lemma 8 that the intractability of the BDH problem implies the THD-IND-CCA security of the THDBM scheme. Combining Lemmas 7 and 8, we obtain Theorem 5.

As mentioned, we describe the THDBM scheme. Actually, THDBM is very similar to IDTHDBM except for some differences in the key/common parameter generation and encryption algorithms. We only describe these two algorithms here.

- $\mathsf{GK}(k, t, n)$: Taking a security parameter k as input, this algorithm generates two groups $\mathcal{G}$ and $\mathcal{F}$ of the same prime order $q \geq 2^k$ and chooses a generator P of $\mathcal{G}$. Then, it specifies the Bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \to \mathcal{F}$ and the following hash functions H_2, H_3, and H_4 such that $\mathsf{H}_2 : \mathcal{F} \to \{0,1\}^l$; $\mathsf{H}_3 : \mathcal{G}^* \times \{0,1\}^l \to \mathcal{G}^*$; $\mathsf{H}_4 : \mathcal{F} \to \mathbb{Z}_q^*$, where l denotes the length of a plaintext. Next, it chooses x uniformly at random from $\mathbb{Z}_q^*$ and computes $Y = xP$. Then, it chooses Q uniformly at random from $\mathcal{G}^*$ and computes $D = xQ$. Note that (Y, D) will be a public/private key pair. Now, given a private key

D, the number of decryption servers n and a threshold parameter t, this algorithm picks $R_1, R_2, \ldots, R_{t-1}$ at random from $\mathcal{G}$ and computes $F(x) = D + \sum_{j=1}^{t-1} x^j R_j$. Then, it computes each server's private key $S_i = F(i)$ for $1 \leq i \leq n$ and verification key $y_i = \hat{e}(S_i, P)$ for $1 \leq i \leq n$. Finally, it outputs a common parameter $cp = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y, Q)$, and sends the verification/private key pair (y_i, S_i) to each decryption server Γ_i for $1 \leq i \leq n$. Upon receiving (y_i, S_i), each decryption server publishes y_i where $1 \leq i \leq n$.

- $\mathsf{E}(cp, m)$: Given a plaintext message $m \in \{0,1\}^l$, this algorithm chooses r uniformly at random from $\mathbb{Z}_q^*$ and computes $d = \hat{e}(Q, Y)$, $\kappa = d^r$ in turn. Then, it computes $U = rP$, $V = \mathsf{H}_2(\kappa) \oplus m$ and $W = r\mathsf{H}_3(U, V)$, and outputs a ciphertext $C = (U, V, W)$.

We now review the chosen ciphertext security notion of (non-ID-based) threshold decryption. First, we denote a generic (t, n) threshold decryption scheme in the non-ID-based setting by "$\mathcal{THD}$". The $\mathcal{THD}$ scheme consists of a key/common parameter generation algorithm GK, an encryption algorithm E, a decryption share generation algorithm D, a decryption share verification algorithm SV, and a share combining algorithm SC.

By running GC, a trusted dealer generates a public key and its matching private key, and shares the private key among a n decryption servers. The dealer also generates (public) verification keys that will be used for share verification. Given the public key, a sender encrypts a plaintext by running E. A user who wants to decrypt a ciphertext gives the ciphertext to the decryption servers requesting decryption shares. The decryption servers then run D to generate corresponding decryption shares. The user can check the validity of the shares by running SV. When the user collects valid decryption shares from at least t servers, the ciphertext can be decrypted by running SC.

We now review the security notion for the threshold decryption scheme against chosen ciphertext attack, which we call "IND-THD-CCA", defined in [112].

Definition 18 (IND-THD-CCA) Let $\mathsf{B}^{\mathsf{CCA}}$ be an attacker assumed to be a probabilistic Turing machine. Suppose that a security parameter k is given to $\mathsf{B}^{\mathsf{CCA}}$ as input. Now, consider the following game in which the attacker $\mathsf{B}^{\mathsf{CCA}}$ interacts with the "Challenger".

Phase 1: $\mathsf{B}^{\mathsf{CCA}}$ corrupts a fixed subset of $t-1$ servers.

Phase 2: The Challenger runs the key/common parameter generation algorithm GK taking a security parameter k as input. The Challenger gives $\mathsf{B}^{\mathsf{CCA}}$ the resulting private keys of the corrupted servers, the public key, the verification key and the common parameter. However, the Challenger keeps the private keys of uncorrupted servers secret from $\mathsf{B}^{\mathsf{CCA}}$.

Phase 3: $\mathsf{B}^{\mathsf{CCA}}$ adaptively interacts with the uncorrupted decryption servers, submitting ciphertexts and obtaining decryption shares.

Phase 4: $\mathsf{B}^{\mathsf{CCA}}$ chooses two equal-length plaintexts (M_0, M_1). If these are given to the encryption algorithm then the Challenger chooses $\beta \in \{0,1\}$ at random and returns a target ciphertext $C^* = \mathsf{E}(cp, pk, M_\beta)$ to $\mathsf{B}^{\mathsf{CCA}}$.

Phase 5: $\mathsf{B}^{\mathsf{CCA}}$ adaptively interacts with the uncorrupted decryption servers, submitting ciphertexts and obtaining decryption shares. However, the target ciphertext C^* is not allowed to query to the decryption servers.

Phase 6: $\mathsf{B}^{\mathsf{CCA}}$ outputs a guess $\tilde{\beta} \in \{0,1\}$.

We define the attacker $\mathsf{B}^{\mathsf{CCA}}$'s success by

$$\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathcal{THD},\mathsf{B}^{\mathsf{CCA}}}(k) = 2 \cdot \Pr[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathcal{THD}}(t_{CCA}, q_D)$ the maximum of the attacker $\mathsf{B}^{\mathsf{CCA}}$'s success over all attackers $\mathsf{B}^{\mathsf{CCA}}$ having running time t_{CCA} and making at most q_D decryption share generation queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The scheme $\mathcal{THD}$ is said to be IND-THD-CCA secure if $\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathcal{THD}}(t_{CCA}, q_D)$ is negligible in k.

We now prove the following lemma.

Lemma 7 *Suppose that an IND-IDTHD-CCA attacker for the* `IDTHDBM` *scheme issues up to q_E private key extraction queries, q_D decryption share generation queries, q_{H_1}, q_{H_2}, q_{H_3}, and q_{H_4} queries to the random oracles H_1, H_2, H_3, and H_4 respectively. Using this attacker as a subroutine, we can construct an IND-THD-CCA attacker for the* `THDBM` *scheme, whose running time and the number of decryption share generation queries and the random oracle queries to H_2, H_3,*

and H_4 *are bounded by* t_{CCA}, q'_D *and* q'_{H_2}, q'_{H_3}, *and* q'_{H_4} *respectively. Concretely, we obtain the following advantage bound:*

$$\frac{1}{q_{\mathsf{H}_1}}\mathbf{Succ}^{\text{IDTHD-IND-CCA}}_{\texttt{IDTHDBM}}(t_{IDCCA}, q_E, q_D, q_{\mathsf{H}_1}, q_{\mathsf{H}_2}, q_{\mathsf{H}_3}, q_{\mathsf{H}_4}) \leq \mathbf{Succ}^{\text{THD-IND-CCA}}_{\texttt{THDBM}}(t_{CCA}, q'_D, q'_{\mathsf{H}_2}, q'_{\mathsf{H}_3}, q'_{\mathsf{H}_4}),$$

where $t_{CCA} = t_{IDCCA} + \max(q_E, q_{\mathsf{H}_1})O(k^3)$, $q'_D = q_D$, $q'_{\mathsf{H}_2} = q_{\mathsf{H}_2}$, $q'_{\mathsf{H}_3} = q_{\mathsf{H}_3}$ *and* $q'_{\mathsf{H}_4} = q_{\mathsf{H}_4}$ *for a security parameter* k. *Here,* t_{IDCCA} *denotes the running time of the IDTHD-IND-CCA attacker.*

Proof. For notational convenience, we assume that the same group parameters $\{\mathcal{G}, q, \hat{e}, P\}$ and security parameter k are given to attackers for `IDTHDBM` and `THDBM`.

Let $\mathsf{A}^{\mathsf{CCA}}$ denote an attacker that defeats the IND-IDTHD-CCA security of the `IDTHDBM` scheme. We assume that $\mathsf{A}^{\mathsf{CCA}}$ has access to the common parameter $cp_{\texttt{IDTHDBM}} = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y_{\text{PKG}})$ of the `IDTHDBM` scheme, where $Y_{\text{PKG}} = x'P$ for random $x' \in \mathbb{Z}_q^*$. We also assume that $\mathsf{A}^{\mathsf{CCA}}$ has access to its decryption servers and a set of verification keys.

Let $\mathsf{B}^{\mathsf{CCA}}$ denote an attacker that defeats the THD-IND-CCA security of the `THDBM` scheme. We assume that $\mathsf{B}^{\mathsf{CCA}}$ has access to the common parameter $cp_{\texttt{THDBM}} = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y, Q)$ of the `THDBM` scheme, where $Y = xP$ for random $x \in \mathbb{Z}_q^*$ and Q has been randomly chosen from $\mathcal{G}$. Also, we assume that $\mathsf{B}^{\mathsf{CCA}}$ has access to its decryption servers and a set of verification keys.

Our aim is to simulate the view of $\mathsf{A}^{\mathsf{CCA}}$ in the real attack game denoted by $\mathbf{G_0}$ until we obtain a game denoted by $\mathbf{G_1}$, which is related to the ability of the attacker $\mathsf{B}^{\mathsf{CCA}}$ to defeat the THD-IND-CCA security of the `THDBM` scheme.

- Game $\mathbf{G_0}$: As mentioned, this game is identical to the real attack game described in Definition 16. We denote by E_0 the event that $\mathsf{A}^{\mathsf{CCA}}$'s output $\bar{\beta} \in \{0,1\}$ is equal to $\beta \in \{0,1\}$ chosen by the Challenger. We use a similar notation E_i for all Games $\mathbf{G}_i$. Since Game $\mathbf{G_1}$ is the same as the real attack game, we have

$$\Pr[E_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\text{IND-IDTHD-CCA}}_{\texttt{IDTHDBM},\mathsf{A}^{\mathsf{CCA}}}(k).$$

- Game $\mathbf{G_1}$: First, we replace Y_{PKG} of $\mathsf{A}^{\mathsf{CCA}}$'s common parameter $cp_{\texttt{IDTHDBM}}$ by Y of $\mathsf{B}^{\mathsf{CCA}}$'s common parameter $cp_{\texttt{THDBM}}$ setting $Y_{\text{PKG}} = Y$. We also

replace $\mathsf{A}^{\mathsf{CCA}}$'s decryption servers by $\mathsf{B}^{\mathsf{CCA}}$'s decryption servers. We then randomly choose an index μ from the range $[1, 2, \ldots, q_{\mathsf{H}_1}]$ where q_{H_1} denotes the maximum number queries to the random oracle H_1 made by $\mathsf{A}^{\mathsf{CCA}}$. By $\mathtt{ID}_\mu$, we denote the μ-th query to the random oracle H_1. We hope $\mathtt{ID}_\mu$ would be a target identity $\mathtt{ID}^*$ that $\mathsf{A}^{\mathsf{CCA}}$ outputs in Phase 4 of the real attack game of IND-IDTHD-CCA described in Definition 16.

Now, we simulate $\mathsf{A}^{\mathsf{CCA}}$'s random oracle H_1, which can be queried at any time during the attack. Whenever H_1 is queried at $\mathtt{ID}$, we perform the following:

- If the query $\mathtt{ID}$ exists in the entry $\langle(\mathtt{ID}, \tau), Q_{\mathtt{ID}}\rangle \in \mathsf{H_1List}$, return $Q_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$. (Note that $\mathsf{H_1List}$ is the "input-output" list for the simulation of H_1.)
- Otherwise, do the following.
  - If $\mathtt{ID} = \mathtt{ID}_\mu$, set $Q_{\mathtt{ID}} = Q$ and return $Q_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$. (Note that Q is from $\mathsf{B}^{\mathsf{CCA}}$'s common parameter $cp_{\mathtt{THDBM}}$.)
  - Else ($\mathtt{ID} \neq \mathtt{ID}_\mu$), do the following.
    - Choose τ uniformly at random from $\mathbb{Z}_q^*$.
    - Compute $Q_{\mathtt{ID}} = \tau P$ and return $Q_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$.

If $\mathsf{A}^{\mathsf{CCA}}$ issues $\mathtt{ID}$ as a private key extraction query, we perform the following:

- If the query $\mathtt{ID}$ exists in the entry $\langle(\mathtt{ID}, Q_{\mathtt{ID}}), \tau)\rangle \in \mathsf{H_1List}$, extract τ from it, compute $D_{\mathtt{ID}} = \tau Y$, and return $D_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$.
- Otherwise, do the following.
  - If $\mathtt{ID} = \mathtt{ID}_\mu$, terminate the whole game. (Note, however, that if $\mathtt{ID}_\mu = \mathtt{ID}^*$, this query is not allowed.)
  - Else ($\mathtt{ID} \neq \mathtt{ID}_\mu$), do the following.
    - Choose τ uniformly at random from $\mathbb{Z}_q^*$.
    - Compute $Q_{\mathtt{ID}} = \tau P$ and save $\langle(\mathtt{ID}, \tau), Q_{\mathtt{ID}}\rangle$ into $\mathsf{H_1List}$.
    - Compute $D_{\mathtt{ID}} = \tau Y$ and return $D_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CCA}}$.

If $\mathsf{A}^{\mathsf{CCA}}$ corrupts $t-1$ decryption servers during the attack, that is, $\mathsf{A}^{\mathsf{CCA}}$ obtains private keys $\{S_i\}_{1 \leq i \leq t-1}$ of corrupted decryption servers, we give them to $\mathsf{B}^{\mathsf{CCA}}$.

If $\mathsf{A}^{\mathsf{CCA}}$ submits a pair of two equal-length plaintexts (M_0, M_1), we give it to $\mathsf{B}^{\mathsf{CCA}}$, then $\mathsf{B}^{\mathsf{CCA}}$ uses (M_0, M_1) as its plaintext-pair to be challenged. $\mathsf{B}^{\mathsf{CCA}}$

queries (M_0, M_1) to its Challenger, and obtains a target ciphertext C^* such that

$$C^* = (U, V, W) = (rP, M_\beta \oplus \mathsf{H}_2(\hat{e}(Q, Y)^r), r\mathsf{H}_3(U, V)),$$

where r and β are chosen uniformly at random from $\mathbb{Z}_q^*$ and $\{0, 1\}$ respectively. We simply return the C^* to $\mathsf{A}^{\mathsf{CCA}}$ as a target ciphertext.

If $\mathsf{A}^{\mathsf{CCA}}$ issues decryption share generation queries after it submits the target identity, $\mathsf{B}^{\mathsf{CCA}}$ uses its decryption servers to answer those queries. Note, however, that $\mathsf{A}^{\mathsf{CCA}}$ is not allowed to query C^* to any of the uncorrupted decryption servers.

Finally, if $\mathsf{A}^{\mathsf{CCA}}$ submits its guess $\tilde{\beta}$, we give it to $\mathsf{B}^{\mathsf{CCA}}$.

Note that due to the randomness of τ in $\mathbb{Z}_q^*$ and Q, the above simulation of the random oracle H_1 is perfect. Note also that $D_{\mathtt{ID}} = \tau Y_{\mathrm{PKG}} = \tau Y = \tau x P = x \tau P = x Q_{\mathtt{ID}}$ and that

$$\begin{aligned} C^* &= (rP, M_\beta \oplus \mathsf{H}_2(\hat{e}(Q_{\mathtt{ID}_\mu}, Y_{\mathrm{PKG}})^r), r\mathsf{H}_3(U, V)) \\ &= (rP, M_\beta \oplus \mathsf{H}_2(\hat{e}(Q_{\mathtt{ID}^*}, Y_{\mathrm{PKG}})^r), r\mathsf{H}_3(U, V)) \\ &= (rP, M_\beta \oplus \mathsf{H}_2(\hat{e}(\mathsf{H}_1(\mathtt{ID}^*), Y_{\mathrm{PKG}})^r), r\mathsf{H}_3(U, V)). \end{aligned}$$

Hence, as long as $\mathtt{ID}_\mu = \mathtt{ID}^*$, the private keys associated with IDs and the target ciphertext C^* that $\mathsf{A}^{\mathsf{CCA}}$ obtain in the simulation are identically distributed as those $\mathsf{A}^{\mathsf{CCA}}$ obtain in the real attack. (Note that if $\mathtt{ID}_\mu = \mathtt{ID}^*$, we do not terminate the game.)

Since μ has been uniformly chosen from $[1, q_{\mathsf{H}_1}]$ and the bit β is uniformly chosen from $\{0, 1\}$, we have

$$\Pr[E_1] - \frac{1}{2} \geq \frac{1}{q_{\mathsf{H}_1}} \Big(\Pr[E_0] - \frac{1}{2} \Big).$$

Thus, by definition of $\Pr[E_0]$ and $\Pr[E_1]$, we obtain

$$\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathtt{THDBM}, \mathsf{B}^{\mathsf{CCA}}}(k) \geq \frac{1}{q_{\mathsf{H}_1}} \mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathtt{IDTHDBM}, \mathsf{A}^{\mathsf{CCA}}}(k).$$

Finally note that the running time t_{CCA} of an arbitrary IND-THD-CCA attacker for the scheme $\mathtt{THDBM}$ is lower-bounded by $t_{IDCCA} + \max(q_E, q_{\mathsf{H}_1}) O(k^3)$. Note also that the number of queries to the random oracles H_2, H_3, H_4 and the decryption

servers made by $\mathsf{B}^{\mathsf{CCA}}$ are the same as the number of those $\mathsf{A}^{\mathsf{CCA}}$ has made. Hence, we obtain the bound in the lemma statement. □

Next, we prove the following lemma.

Lemma 8 *Suppose that an IND-THD-CCA attacker for the* THDBM *scheme issues up to q_D decryption share generation queries, q_{H_2}, q_{H_3}, and q_{H_4} queries to the random oracles H_2, H_3, and H_4 respectively. Using this attacker as a subroutine, we can construct a BDH attacker for the group $\mathcal{G}$, whose running time is bounded by t_{BDH}. Concretely, we obtain the following advantage bound:*

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathtt{THDBM}}(t, q_{\mathsf{H}_2}, q_{\mathsf{H}_3}, q_{\mathsf{H}_4} q_D) \leq \mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G}}(t_{\mathrm{BDH}}) + \frac{q_D + q_D q_{\mathsf{H}_4}}{2^k},$$

where $t_{\mathrm{BDH}} = t_{\mathrm{CCA}} + q_{\mathsf{H}_2} + q_{\mathsf{H}_3} O(k^3) + q_{\mathsf{H}_4} q_D O(k^3)$ for a security parameter k.

Proof. For notational convenience, we assume that the same group parameters $\{\mathcal{G}, q, \hat{e}, P\}$ and security parameter k are given to attackers for THDBM and the BDH problem.

Let $\mathsf{B}^{\mathsf{CCA}}$ be an attacker that defeats the THD-IND-CCA security of the THDBM scheme. Let $\mathsf{A}^{\mathsf{BDH}}$ be an attacker for the BDH problem. Suppose that ($\mathcal{G}$, q, $\hat{e}$, P, aP, bP, cP) is given to $\mathsf{A}^{\mathsf{BDH}}$. Also, suppose that the (same) security parameter k is given to $\mathsf{B}^{\mathsf{CCA}}$.

We start with Game $\mathbf{G_0}$ which is the same as the real attack game associated with $\mathsf{B}^{\mathsf{CCA}}$. Then, we modify this game until we completely simulate the view of $\mathsf{B}^{\mathsf{CCA}}$ and obtain a game in which $\mathsf{A}^{\mathsf{BDH}}$ is able to solve the BDH problem.

- Game $\mathbf{G_0}$: This game is actually the same as the real attack game. However, we repeat it for cleaning up notations.

  First, we run the key/common parameter generation algorithm of the THDBM scheme on input a security parameter k, a threshold parameter t and a number of decryption servers n. We give $\mathsf{B}^{\mathsf{CCA}}$ the resulting common parameter $cp_{\mathtt{THDBM}} = (\mathcal{G}, q, \hat{e}, P, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y, Q)$ where $Y = xP$ for random $x \in \mathbb{Z}_q^*$ and the set of verification keys $\{y_i\}$, where $1 \leq i \leq n$. But we keep the private key $D = xQ$ as secret.

  If $\mathsf{B}^{\mathsf{CCA}}$ submits a pair of plaintexts (M_0, M_1), we choose a bit β uniformly at random and create a target ciphertext $C^* = (U^*, V^*, W^*)$ as follows.

$$U^* = r^* P, V^* = H_2^* \oplus M_\beta, \text{ and } W^* = r^* H_3^*,$$

where $\kappa^* = \hat{e}(Q, Y)^{r^*}$ for random $r^* \in \mathbb{Z}_q^*$, $H_2^* = \mathsf{H}_2(\kappa^*)$ and $H_3^* = \mathsf{H}_3(U^*, V^*)$.

Once all the decryption servers are set up, $\mathsf{B}^{\mathsf{CCA}}$ can issue decryption share generation queries at its will. We denote those queries by $C = (U, V, W)$. Note that C is different from the target ciphertext C^*.

On input C^*, $\mathsf{B}^{\mathsf{CCA}}$ outputs $\tilde{\beta}$. We denote by E_0 the event $\tilde{\beta} = \beta$ and use a similar notation E_i for all $\mathbf{G}_i$. Since game $\mathbf{G_0}$ is the same as the real attack game, we have

$$\Pr[E_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{THD-IND-CCA}}_{\mathtt{THDBM},\mathsf{B}^{\mathsf{CCA}}}(k).$$

- Game $\mathbf{G_1}$: First, we replace replace Y and Q in $cp_{\mathtt{THDBM}}$ by bP and cP respectively, all of which are given to $\mathsf{A}^{\mathsf{BDH}}$. We denote bP and cP by Y_{BDH} and Q_{BDH} respectively. Now, we assume that a subset of $t-1$ decryption servers have been corrupted without loss of generality. Let $\Phi' = \{0, 1, \ldots, t-1\}$. Then, we choose $S_1, S_2, \ldots, S_{t-1}$ uniformly at random from $\mathcal{G}$ and compute

$$y_i = \hat{e}(Q_{\mathrm{BDH}}, Y_{\mathrm{BDH}})^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, P)^{c_{ij}^{\Phi'}},$$

where $t \le i \le n$ and $c_{ij}^{\Phi'}$ denotes a Lagrange coefficient with respect to the set Φ'. We send S_i where $1 \le i \le t-1$ to each of the corrupted servers and send y_i where $t \le i \le n$ to each of the uncorrupted decryption servers. Then, $\mathsf{B}^{\mathsf{CCA}}$ obtains access to $\{S_i\}$ and $\{y_i\}$.

Now, we modify the target ciphertext $C^* = (U^*, V^*, W^*)$ as follows. First, we choose κ^+ uniformly at random from $\mathcal{F}$ and replace κ^* by κ^+. We also choose H_2^+ uniformly at random from $\{0,1\}^l$, replace H_2^* by H_2^+ and V^* by $V^+ = H_2^+ \oplus M_\beta$. Accordingly, whenever the random oracle H_2 is queried at κ^+, we respond with H_2^+.

Summing up, we obtain a new challenge ciphertext denoted by C_+^* such that $C_+^* = (U^*, V^+, W^*)$, where $V^+ = H_2^+ \oplus M_\beta$ and $H_2^+ = \mathsf{H}_2(\kappa^+)$ for random $\kappa^+ \in \mathcal{F}$.

Note that the attacker $\mathsf{B}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_0}$ and Game $\mathbf{G_1}$, since we have replaced one set of random variables by another set of random variables which is different, yet has the same distribution.

Thus, we have

$$\Pr[E_1] = \Pr[E_0].$$

- Game $\mathbf{G_2}$: In this game, we restore the queries to the random oracle H_2. That is, if H_2 is queried at κ^+, we do not respond with H_2^+ any more but respond with an answer from the random oracle H_2 instead. We assume that this rule applies to all the forthcoming games.

  By the above rule, κ^+ and H_2^+ are used only in the target ciphertext C_+^*. Accordingly, the distribution of input to $\mathsf{B}^{\mathsf{CCA}}$ does not depend on β. Hence, we get $\Pr[E_2] = 1/2$.

  Note that Game $\mathbf{G_2}$ and Game $\mathbf{G_1}$ may differ if the random oracle H_1 is queried at κ^*. Let AskH_{2_2} denotes the event that, in game $\mathbf{G_2}$, H_2 is queried at κ^*. We will use the same notation AskH_{2_i} to denote such events in all other games.

  Now, we have

$$|\Pr[E_2] - \Pr[E_1]| \leq \Pr[\mathsf{AskH}_{2_2}].$$

- Game $\mathbf{G_3}$: In this game, we further modify the target ciphertext $C_+^* = (U^*, V^+, W^*)$. First, we replace U^* by aP. We keep $V^+ (= H_2^+ \oplus M_\beta = \mathsf{H}_2(\kappa^+) \oplus M_\beta)$ as it is, but define κ^+ as the BDH key $\hat{e}(P,P)^{abc}$. Then, we choose s^+ uniformly at random from $\mathbb{Z}_q^*$, compute s^+aP and replace W^* by s^+aP. Finally, we modify the computation of the random oracle H_3 as follows. Whenever H_3 is queried at (aP, V^+), we compute $H_3^+ = s^+P$ and respond with H_3^+. Namely, we set $H_3^+ = \mathsf{H}_2(aP, V^+)$.

  Summing up, we have obtained a new target ciphertext denoted by C_{BDH}^* $= (U_{\mathrm{BDH}}, V_{\mathrm{BDH}}, W_{\mathrm{BDH}})$ such that

$$U_{\mathrm{BDH}} = aP;\ V_{\mathrm{BDH}} = V^+;\ W_{\mathrm{BDH}} = s^+aP,$$

  where $V^+ = \mathsf{H}_2(\hat{e}(P,P)^{abc}) \oplus M_\beta$. Moreover, we have $\mathsf{H}_3(U_{\mathrm{BDH}}, V_{\mathrm{BDH}}) = H_3^+ = s^+P$.

  Note that we have replaced one set of random variables $\{U^*, W^*\}$ by another set of random variables $\{aP, s^+aP\}$ which is different, yet has the same distribution. Note also that C_{BDH}^* is a valid ciphertext since $\hat{e}(P, W_{\mathrm{BDH}}) = \hat{e}(U_{\mathrm{BDH}}, H_3^+)$ by the construction of H_3^+ and W_{BDH}. Hence, the attacker $\mathsf{B}^{\mathsf{CCA}}$'s view has the same distribution in both Game $\mathbf{G_2}$ and Game $\mathbf{G_3}$,

and we have

$$\Pr[\mathsf{AskH}_{2_3}] = \Pr[\mathsf{AskH}_{2_2}].$$

- Game $\mathbf{G_4}$: In this game, we modify the random oracle H_3. Note that we have already dealt with the simulation of the random oracles H_3 appeared in the target ciphertext C^*_{BDH}, namely, the case when H_3 is queried at $(U_{\mathrm{BDH}}, V_{\mathrm{BDH}})$. In the following, we deal with the rest of simulation.

  Whenever H_3 is queried at $(U, V) \neq (U_{\mathrm{BDH}}, V_{\mathrm{BDH}})$, we choose s uniformly at random from $\mathbb{Z}^*_q$, computes $H_3 = sY$ and respond with H_3. Let $\mathsf{H_3List}$ be a list of all "input-output" pairs of the random oracle H_3. Specifically, $\mathsf{H_3List}$ consists of the pairs $\langle (U, V), H_3 \rangle$ where $H_3 = \mathsf{H}_3(U, V) = sY$. Notice that this list grows as $\mathsf{B}^{\mathsf{CCA}}$'s attack proceeds.

  Because H_3 is assumed to be a random oracle, the above generation of the outputs of H_3 perfectly simulates the real oracle. Hence, $\mathsf{B}^{\mathsf{CCA}}$'s view in this game remains the same as that in the previous game. Hence, we have

$$\Pr[\mathsf{AskH}_{2_4}] = \Pr[\mathsf{AskH}_{2_3}].$$

  Note that the decryption oracle has been regarded as perfect up to this game. The rest of games will deal with simulation of the decryption oracle.

- Game $\mathbf{G_5}$: In this game, we make the decryption oracle reject all ciphertexts $C = (U, V, W)$ such that $H_3 = \mathsf{H}_3(U, V)$ has *not* been queried. If C is a valid ciphertext while $\mathsf{H}_3(U, V)$ has *not* been queried, $\mathsf{B}^{\mathsf{CCA}}$'s view in Game $\mathbf{G_5}$ and Game $\mathbf{G_4}$ may differ.

  Note that if a ciphertext C is valid then it should be the case that $\hat{e}(P, W) = \hat{e}(U, H_3)$. However, since we have assumed that H_3 has not been queried in this game, the above equality holds with probability at most $1/2^k$ since output of the simulated random oracle H_3 is uniformly distributed in $\mathcal{G}$. Adding up all the decryption queries up to q_D, we have

$$|\Pr[\mathsf{AskH}_{2_5}] - \Pr[\mathsf{AskH}_{2_4}]| \leq \frac{q_D}{2^k}.$$

- Game $\mathbf{G_6}$: In this game, we modify the decryption oracle in the previous game to yield a decryption oracle simulator which decrypts a submitted decryption query $C = (U, V, W)$ without the private key. Note that the case when $\mathsf{H}_3(U, V)$ has not been queried are excluded in this game since

it was already dealt with in the previous game. Hence, we assume that $\mathsf{H}_3(U, V)$ has been queried at some point.

Now we describe the complete specification of the decryption oracle simulator. On input a ciphertext $C = (U, V, W)$, the decryption oracle simulator works as follows.

- Extract $\langle (U, V), H_3 \rangle$ from $\mathsf{H_3List}$.
- If $\hat{e}(P, W) = \hat{e}(U, H_3)$
  - Compute $K = (1/s)W$. (Note here that $(1/s)W = (1/s)rsY = rY = rxP$.)
  - Compute $\kappa = \hat{e}(Q, K)$
  - For $t \leq i \leq n$, compute $\kappa_i = \kappa^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, U)^{c_{ij}^{\Phi'}}$.
  - Return κ_i.
- Else reject C.

Note in the above construction that

$$\begin{aligned}
\kappa_i &= \kappa^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, U)^{c_{ij}^{\Phi'}} = \hat{e}(Q, K)^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, U)^{c_{ij}^{\Phi'}} \\
&= \hat{e}(Q, rxP)^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, rP)^{c_{ij}^{\Phi'}} = \hat{e}(Q, xP)^{rc_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, P)^{rc_{ij}^{\Phi'}} \\
&= \left(\hat{e}(Q, Y)^{c_{i0}^{\Phi'}} \prod_{j=1}^{t-1} \hat{e}(S_j, P)^{c_{ij}^{\Phi'}} \right)^r = y_i^r.
\end{aligned}$$

Hence, κ_i is a correct i-th share of the BDH key $\kappa = \hat{e}(Q, Y)^r$. However, we need more efforts to simulate a decryption share δ_i containing κ_i completely. This can be done as follows.

First, we simulate the random oracle H_4 in a classical way. That is, if H_4 is queried, we choose H_4 uniformly at random from $\mathbb{Z}_q^*$ and respond with it. As usual, we maintain an "input-output" list $\mathsf{H_4List}$ for H_4 whose entry is of the form $\langle (\kappa, \tilde{\kappa}, \tilde{y}), H_4 \rangle$. Next, we choose L_i and λ_i uniformly at random from $\mathcal{G}$ and $\mathbb{Z}_q^*$ respectively, and compute $\tilde{\kappa_i} = \hat{e}(L_i, U)/\kappa_i^{\lambda_i}$ and $\tilde{y_i} = \hat{e}(L_i, P)/y_i^{\lambda_i}$. Then, we set $\lambda_i = \mathsf{H}_4(\kappa_i, \tilde{\kappa_i}, \tilde{y_i})$. Finally, we check whether there exists an entry $\langle (\kappa, \tilde{\kappa}, \tilde{y}), H_4 \rangle$ in $\mathsf{H_4List}$ satisfying $H_4 = \lambda_i$ but $(\kappa, \tilde{\kappa}, \tilde{y}) \neq (\kappa_i, \tilde{\kappa_i}, \tilde{y_i})$. If such entry exists then we return "*Abort*" message to $\mathsf{B}^{\mathsf{CCA}}$. Otherwise, we return the simulated value $\delta_i = (i, \kappa_i, \tilde{\kappa_i}, \tilde{y_i}, L_i)$ to

$\mathsf{B}^{\mathsf{CCA}}$ as a decryption share corresponding to C and save $\langle(\kappa_i, \tilde{\kappa_i}, \tilde{y_i},), \lambda_i\rangle$ to $\mathsf{H}_4\mathsf{List}$.

Since H_3 are assumed to have already been queried in this game (i.e., these case were already dealt with in the previous game), the above simulated decryption share generation server perfectly simulates the real one except the error (collision) in the simulation of H_4 occurs. Note that this happens with probability $q_{\mathsf{H}_4}/2^k$, considering up to q_{H_4} queries to H_4. Adding up all the decryption queries up to q_D, we have

$$|\Pr[\mathsf{AskH}_{2_6}] - \Pr[\mathsf{AskH}_{2_5}]| \leq \frac{q_D q_{\mathsf{H}_4}}{2^k}.$$

Now, recall that the target ciphertext used so far is C^*_{BDH} that constructed in Game $\mathbf{G_3}$. Accordingly, AskH_{2_6} denotes an event that the BDH key $\hat{e}(P,P)^{abc}$ has been queried to the random oracle H_2. Note also that we have used the easiness of the DDH problem in the group $\mathcal{G}$ to simulate the decryption oracle.

Therefore, at this stage, $\mathsf{A}^{\mathsf{BDH}}$ can solve the BDH problem by outputting the queries to the random oracle H_2. That is, we have

$$\Pr[\mathsf{AskH}_{2_6}] \leq \mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{BDH}}}(k).$$

Thus, putting all the bounds we have obtained in each game together, we have

$$\begin{aligned}
\frac{1}{2}\mathbf{Succ}^{\mathrm{IND-THD-CCA}}_{\mathtt{THDBM},\mathsf{B}^{\mathsf{CCA}}}(k) &= |\Pr[E_0] - \Pr[E_2]| \leq \Pr[\mathsf{AskH}_{2_2}] \leq \Pr[\mathsf{AskH}_{2_5}] + \frac{q_D}{2^k} \\
&\leq \frac{q_D}{2^k} + \Pr[\mathsf{AskH}_{2_6}] + \frac{q_D q_{\mathsf{H}_4}}{2^k} \\
&\leq \frac{q_D + q_D q_{\mathsf{H}_4}}{2^k} + \mathbf{Succ}^{\mathrm{BDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{BDH}}}(k).
\end{aligned}$$

Considering the running time t_{BDH} and queries of an arbitrary BDH-attacker for the group $\mathcal{G}$, we obtain the bound in the lemma statement. □

5.6 Application to Mediated ID-Based Encryption Schemes

5.6.1 Security Issues in Mediated ID-Based Encryption

The main motivation of mediated cryptography [26] is to revoke a user's privilege to perform cryptographic operations such as decrypting ciphertexts or signing messages *instantaneously*. In [26], Boneh et al. constructed the first mediated encryption and signature schemes using the RSA primitive. Their idea was to split a user's private key into two parts and give one piece to the on-line Security Mediator (SEM) and the other to the user. To decrypt or sign, the user must acquire a message-specific token which is associated with the SEM part of private key from the SEM. As a result, revocation is achieved by instructing the SEM not to issue tokens for the user.

Recently, the problem of realizing mediated encryption in the ID-based setting was considered by Ding and Tsudik [46]. They proposed an ID-based mediated encryption scheme based on RSA-OAEP [18]. Although their scheme offers good performance and practicality, it has a drawback which stems from the fact that a common RSA modulus is used for all the users within the system and hence anyone obtained a single private/public key pair can factor the modulus by compromising the SEM. Therefore, to guarantee the security of Ding and Tsudik's scheme, one should assume that the SEM's private key must be protected throughout the life of the system.

As an alternative to Ding and Tsudik's scheme, Libert and Quisquater [75] proposed a new mediated ID-based encryption scheme. Due to its structural property (mainly because Boneh and Franklin's ID-based encryption scheme is used), their scheme does not suffer from the "common modulus" problem of Ding and Tsudik's scheme. That is, a compromise of the SEM's private key does not lead to a break of the whole system. In contrast to this positive result, Libert and Quisquater, however, observed that *even though the SEM's private key is protected*, their scheme as well as Ding and Tsudik's scheme are not secure against "inside attack" in which the attacker who possesses the user part of private key conducts chosen ciphertext attack. As a result, it should be strictly assumed in those schemes that users' private keys must be protected to ensure chosen ciphertext security. In practice, *this assumption is fairly strong* in that there may be more chance for users to compromise their private keys than the SEM does since the SEM is usually assumed to be a trusted entity configured by a system administrator.

According to Libert and Quisquater [75], the reason why the mediated schemes proposed so far (including their scheme) are not secure against inside attack is that in the SEM architecture of those schemes, there is no mechanism for checking the validity of a ciphertext before generating a token from it. (That is, "publicly checkable" validity test for a ciphertext is missing.) To get an intuition for this, let us examine Libert and Quisquater's scheme which can be described as follows. In the setup stage, the common parameters including the PKG's public key $Y_{\mathrm{PKG}} = xP$ where x is the master key and hash functions H_1, H_2, H_3, and H_4 are generated. In the key generation stage, the PKG computes $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = xQ_{\mathtt{ID}}$ on receiving a user's identity $\mathtt{ID}$. Then, it chooses a random point $D_{\mathtt{ID},user}$ from $\mathcal{G}^*$ and computes $D_{\mathtt{ID},sem} = D_{\mathtt{ID}} - D_{\mathtt{ID},user}$. The PKG gives the partial private key $D_{\mathtt{ID},user}$ to the user and $D_{\mathtt{ID},sem}$ to the SEM. Given the user's identity $\mathtt{ID}$ and the common parameter, a sender can encrypt a message $M \in \{0,1\}^l$ by computing $C = (U, V, W)$ where $U = rP$, $V = \sigma \oplus \mathsf{H}_2(\hat{e}(Y_{PKG}, Q_{\mathtt{ID}})^r)$, $W = M \oplus \mathsf{H}_4(\sigma)$, and $r = \mathsf{H}_3(\sigma, M) \in \mathbb{Z}_q^*$ for random $\sigma \in \{0,1\}^l$. On receiving $C = (U, V, W)$, the user forwards it to the SEM. Then, the SEM and the user perform the following in parallel.

- SEM (We call this procedure "SEM oracle"): Check if the user's identity $\mathtt{ID}$ is revoked. If it is, return "$\mathtt{ID}$ *Revoked*". Otherwise, compute $g_{sem} = \hat{e}(U, D_{\mathtt{ID},sem})$ and send it to the user.
- User (We call this procedure "User oracle"): Compute $g_{user} = \hat{e}(U, D_{\mathtt{ID},user})$. When receiving g_{sem} from the SEM, compute $g = g_{sem} g_{user}$. Then, compute $\sigma = V \oplus \mathsf{H}_2(g)$ and $M = W \oplus \mathsf{H}_4(\sigma)$. Finally, check $U = r'P$ for $r' = \mathsf{H}_3(\sigma, M)$. If it is, return M, otherwise, return "*Reject*".

Now, let us assume that a "strong" attacker has a target ciphertext $C^* = (U^*, V^*, W^*)$ which encrypts a message M_β where β is chosen at random from $\{0,1\}$, a target identity $\mathtt{ID}^*$, and a private key $D_{\mathtt{ID}^*,user}$ associated with $\mathtt{ID}^*$. The attacker can easily defeat the indistinguishability of C^* by conducting a simple chosen ciphertext attack as follows. The attacker just changes C^* into $C' = (U^*, V^*, W')$ for $W' \neq W^*$, and queries this to the SEM oracle. Upon receiving C', the SEM oracle will simply return $g^*_{sem} = \hat{e}(U^*, D_{\mathtt{ID}^*,sem})$. The attacker then computes $g^* = g^*_{sem} g^*_{user}$ where $g^*_{user} = \hat{e}(U^*, D_{\mathtt{ID}^*,user})$ using the user's private key $D_{\mathtt{ID}^*,user}$. Obviously, this problem is caused by the fact that the validity of queried ciphertexts is not checked in the SEM oracle.

Libert and Quisquater remained the removal of this drawback as an open problem, only providing a proof that their scheme is secure against chosen ciphertext attack

in a *weaker* sense – attackers are *not* allowed to obtain the user part of private key.

However, in the following section, we present a new mediated ID-based encryption scheme which is secure against ciphertext attack in a *strong* sense, that is, secure against chosen ciphertext attack conducted by the stronger attacker who obtains the user part of private key.

5.6.2 Description of Our Mediated ID-Based Encryption Scheme

In what follows, we describe our mediated ID-based encryption scheme "MIDEBM", which is based on the IDTHDBM scheme described in Section 5.5.2. MIDEBM consists of the following algorithms.

- A randomized key/common parameter generation algorithm $\mathsf{GC}(k)$: Given a security parameter k, the PKG runs the key generation algorithm of IDTHDBM. The output of this algorithm $cp = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_4, Y_{\mathrm{PKG}})$ is as defined in the description of IDTHDBM. Note that cp is given to all interested parties while the master key x is kept secret within the PKG.
- A private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: This algorithm is run by the PKG on receiving a private key extraction query from any user who wants to extract a private key that matches to an identity ID.
  - Given ID, compute $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = sQ_{\mathtt{ID}}$.
  - Output $D_{\mathtt{ID}}$.
- A randomized private key distribution algorithm $\mathsf{DK}(cp, D_{\mathtt{ID}})$: Given $D_{\mathtt{ID}}$ which is a private key associated with an identity ID, the PKG splits $D_{\mathtt{ID}}$ using $(2, 2)$ secret-sharing technique as follows.
  - Pick R at random from $\mathcal{G}^*$ and construct $F(u) = D_{\mathtt{ID}} + uR$ for $u \in \{0\} \cup \mathbb{N}$.
  - Compute $D_{\mathtt{ID},sem} = F(1)$ and $D_{\mathtt{ID},user} = F(2)$.

  The PKG gives $D_{\mathtt{ID},sem}$ to the SEM and $D_{\mathtt{ID},user}$ to the user.

- A randomized encryption algorithm $\mathsf{E}(cp, \mathtt{ID}, M)$: Given a plaintext $M \in \{0,1\}^l$ and a user's identity $\mathtt{ID}$, a user creates a ciphertext $C = (U, V, W)$ such that

$$U = rP; V = \mathsf{H}_2(\kappa) \oplus M; W = r\mathsf{H}_3(U, V),$$

  where $\kappa = \hat{e}(\mathsf{H}_1(\mathtt{ID}), Y_{\mathrm{PKG}})^r$ for random $r \in \mathbb{Z}_q^*$.

- A decryption algorithm $\mathsf{D}(cp, D_{\mathtt{ID},sem}, D_{\mathtt{ID},user}, C)$: When receiving $C = (U, V, W)$, a user forwards it to the SEM. The SEM and the user perform the following in parallel.

  - SEM (We call this procedure "SEM oracle"):
    1. Check if the user's identity $\mathtt{ID}$ is revoked. If it is, return "$\mathtt{ID}$ *Revoked*".
    2. Otherwise, do the following:
       * Compute $H_3 = \mathsf{H}_3(U, V)$ and check if $\hat{e}(P, W) = \hat{e}(U, H_3)$. If C has passed this test, compute $\kappa_{sem} = \hat{e}(D_{\mathtt{ID},sem}, U)$ and send $\delta_{\mathtt{ID},sem,C} = (sem, \kappa_{sem})$ to the user. Otherwise, send $\delta_{\mathtt{ID},sem,C} = (sem,$ "*Invalid Ciphertext*") to the user.
  - User (We call this procedure "User oracle"):
    1. Compute $H_3 = \mathsf{H}_3(U, V)$ and check if $\hat{e}(P, W) = \hat{e}(U, H_3)$. If C has passed this test, compute $\kappa_{user} = \hat{e}(D_{\mathtt{ID},user}, U)$. Otherwise, return "*Reject*" and terminate.
    2. Get $\delta_{\mathtt{ID},sem,C}$ from the SEM and do the following:
       * If $\delta_{\mathtt{ID},sem,C}$ is of the form $(sem,$ "*Invalid Ciphertext*"), return "*Reject*" and terminate. Otherwise, compute $\kappa = \kappa_{sem}^{c_{01}^{\Phi}} \kappa_{user}^{c_{02}^{\Phi}}$ where c_{01}^{Φ} and c_{02}^{Φ} denote the Lagrange coefficients for the set $\Phi = \{1, 2\}$ and $M = \mathsf{H}_2(\kappa) \oplus V$, and return M.

Notice that in the SEM oracle of our scheme, the validity of a ciphertext is checked before generating a token in the same way as the decryption share generation algorithm of `IDTHDBM` does.

5.6.3 Security Analysis of Our Mediated ID-Based Encryption Scheme

In this section, we show that the chosen ciphertext security of the above scheme against the strong attacker that obtains the user part of private key is relative to the IND-IDTHD-CCA (Definition 16) security of the $(2, 2)$-`IDTHDBM` scheme.

To begin with, we define IND-mID-sCCA (indistinguishability of mediated ID-based encryption against strong chosen ciphertext attack), which is similar to IND-mID-wCCA ("w" stands for "weak") defined in [75] but assumes the stronger attacker that can corrupt users to get their private keys.

Definition 19 (IND-mID-sCCA) Let $\mathsf{A}^{\mathsf{CCA}'}$ be an attacker that defeats the IND-mID-sCCA security of an mediated ID-based encryption scheme $\mathcal{MIDE}$ which consists of GK, EX, DK, E, and D. (For details of these algorithms, readers are referred to [46], [75] or the description of `MIDEBM` given in Section 5.6.2.) We assume that $\mathsf{A}^{\mathsf{CCA}'}$ is a probabilistic Turing machine taking a security parameter k as input. Consider the following game in which the attacker $\mathsf{A}^{\mathsf{CCA}'}$ interacts with the "Challenger".

> **Phase 1**: The Challenger runs the Setup algorithm taking a security parameter k. The Challenger then gives the common parameter to $\mathsf{A}^{\mathsf{CCA}'}$.
>
> **Phase 2**: Having obtained the common parameter, $\mathsf{A}^{\mathsf{CCA}'}$ issues the following queries.
>
> - "User key extraction" query `ID`: On receiving this query, the Challenger runs the Keygen algorithm to obtain the user part of private key and sends it to $\mathsf{A}^{\mathsf{CCA}'}$.
> - "SEM key extraction" query `ID`: On receiving this query, the Challenger runs the Keygen algorithm to obtain the SEM part of private key and sends it to $\mathsf{A}^{\mathsf{CCA}'}$.
> - "SEM oracle" query $(\mathtt{ID}, C)$: On receiving this query, the Challenger runs the Keygen algorithm to obtain a SEM part of private key. Taking the resulting private key as input, the Challenger runs the SEM oracle in the Decrypt algorithm to obtain a decryption token for C and sends it to $\mathsf{A}^{\mathsf{CCA}'}$.
> - "User oracle" query $(\mathtt{ID}, C)$: On receiving this query, the Challenger runs the Keygen algorithm to obtain a User part of private key. Taking

the resulting private key as input, the Challenger runs the User oracle in the Decrypt algorithm to obtain a decryption token for C and sends it to $\mathsf{A}^{\mathsf{CCA}'}$.

Phase 3: $\mathsf{A}^{\mathsf{CCA}'}$ selects two equal-length plaintexts (M_0, M_1) and a target identity $\mathtt{ID}^*$ which was not queried before. On receiving (M_0, M_1) and $\mathtt{ID}^*$, the Challenger runs the Keygen algorithm to obtain User and SEM parts of the private key associated with $\mathtt{ID}^*$. The Challenger then chooses $\beta \in \{0, 1\}$ at random and creates a target ciphertext C^* by encrypting M_β under the target identity $\mathtt{ID}^*$. The Challenger gives the target ciphertext and *the User part of the private key* to $\mathsf{A}^{\mathsf{CCA}'}$.

Phase 4: $\mathsf{A}^{\mathsf{CCA}'}$ continues to issue "User key extraction" query $\mathtt{ID} \neq \mathtt{ID}^*$, "SEM key extraction" query $\mathtt{ID} \neq \mathtt{ID}^*$, "SEM oracle query" $(\mathtt{ID}, C) \neq (\mathtt{ID}^*, C^*)$, and "User oracle" query $(\mathtt{ID}, C) \neq (\mathtt{ID}^*, C^*)$. The details of these queries are as described in Phase 2.

Phase 5: $\mathsf{A}^{\mathsf{CCA}'}$ outputs a guess $\tilde{\beta} \in \{0, 1\}$.

We define the attacker $\mathsf{A}^{\mathsf{CCA}'}$'s success by

$$\mathbf{Succ}^{\mathrm{IND-mID-sCCA}}_{\mathcal{MIDE},\mathsf{A}^{\mathsf{CCA}'}}(k) = 2 \cdot \Pr[\tilde{\beta} = \beta] - 1.$$

We denote by $\mathbf{Succ}^{\mathrm{IND-mID-sCCA}}_{\mathcal{MIDE}}(t_{CCA}, q_{E,user}, q_{E,sem}, q_{D,sem}, q_{D,user})$ the maximum of the attacker $\mathsf{A}^{\mathsf{CCA}'}$'s success over all attackers $\mathsf{A}^{\mathsf{CCA}'}$ having running time t_{CCA} and making at most $q_{E,user}$ "User key extraction" queries, $q_{E,sem}$ "SEM key extraction" queries, $q_{D,sem}$ "SEM oracle" queries, and $q_{D,user}$ "User oracle" queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The mediated ID-based encryption scheme $\mathcal{MIDE}$ is said to be IND-mID-sCCA secure if $\mathbf{Succ}^{\mathrm{IND-mID-sCCA}}_{\mathcal{MIDE}}(t_{CCA}, q_{E,user}, q_{E,sem}, q_{D,sem}, q_{D,user})$ is negligible in k.

We now state and prove the following theorem which implies if the `IDTHDBM` scheme is IND-IDTHD-CCA secure then the `MIDEBM` scheme is IND-mID-sCCA secure.

Theorem 6 *Suppose that an IND-mID-sCCA attacker for the* `MIDEBM` *scheme issues up to $q_{E,user}$ "User key extraction" queries, $q_{E,sem}$ "SEM key extraction" queries, $q_{D,sem}$ "SEM oracle", and $q_{D,user}$ "User oracle" queries. Using this attacker as a subroutine, we can construct an IND-IDTHD-CCA attacker for the*

IDTHDBM *scheme with* $(t, n) = (2, 2)$*, whose running time and the number of private key extraction and decryption share generation queries are bounded by* t_{IDCCA}*,* q_E*, and* q_D *respectively. Concretely, we obtain the following advantage bound:*

$$\begin{aligned}\mathbf{Succ}^{\text{IND-mID-sCCA}}_{\texttt{MIDEBM}}(t_{CCA,E,user}\,, q_{E,sem}, q_{D,sem}, q_{D,user}) \\ \leq \mathbf{Succ}^{\text{IND-IDTHD-CCA}}_{\texttt{IDTHDBM}}(t_{IDCCA}, q_E, q_D),\end{aligned}$$

where $t_{IDCCA} = t_{CCA} + \max(q_{E,user}, q_{E,sem}, q_{D,sem}, q_{D,user})O(k^3)$*,* $q_E = O(1)(q_{E,user} + q_{E,sem} + q_{D,sem} + q_{D,user})$*,* $q_D = O(1)(q_{D,sem} + q_{D,user})$ *for a security parameter* k*. Here,* t_{CCA} *denotes the running time of the ID-mID-sCCA attacker.*

Proof. For notational convenience, we assume that the same group parameter $cp = \{\mathcal{G}, q, \hat{e}, P, Y_{\text{PKG}}\}$ where $Y_{\text{PKG}} = xP$ and security parameter k are given to attackers for MIDEBM and IDTHDBM.

Let $\mathsf{A}^{\mathsf{CCA}'}$ denote an attacker that defeats the IND-mID-sCCA security of the MIDEBM scheme. Let $\mathsf{A}^{\mathsf{CCA}}$ denote an attacker that defeats the IND-IDTHD-CCA security of the IDTHDBM scheme with $(t, n) = (2, 2)$.

Our aim is to simulate the view of $\mathsf{A}^{\mathsf{CCA}'}$ in the real attack game denoted by $\mathbf{G_0}$ until we obtain a game denoted by $\mathbf{G_1}$, which is related to the ability of the attacker $\mathsf{A}^{\mathsf{CCA}}$ to defeat the IND-IDTHD-CCA security of the IDTHDBM scheme.

- Game $\mathbf{G_0}$: As mentioned, this game is identical to the real attack game described in Definition 19. We denote by E_0 the event that $\mathsf{A}^{\mathsf{CCA}'}$'s output $\bar{\beta} \in \{0, 1\}$ is equal to $\beta \in \{0, 1\}$ chosen by the Challenger. We use a similar notation E_i for all Games $\mathbf{G}_i$. Since Game $\mathbf{G_1}$ is the same as the real attack game, we have

$$\Pr[E_0] = \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\text{IND-mID-sCCA}}_{\texttt{MIDEBM},\mathsf{A}^{\mathsf{CCA}}}(k).$$

- Game $\mathbf{G_1}$: First, we give $\mathsf{A}^{\mathsf{CCA}}$'s common parameter to $\mathsf{A}^{\mathsf{CCA}'}$. We then deal with the simulation of $\mathsf{A}^{\mathsf{CCA}'}$'s view in Phase 2 of the real attack game as follows.

  On receiving $\mathsf{A}^{\mathsf{CCA}'}$'s "User key extraction" queries, each of which consists of ID, we perform the following:

  - Search UserKeyList which consists of ⟨identity, corresponding user part of private key⟩ pairs for an entry that is matched against ID.

    * If there exists such an entry, extract a corresponding user part of private key and return it to $\mathsf{A}^{\mathsf{CCA}\prime}$ as an answer.
    * Otherwise, do the following:
        · Forward ID as a "private key extraction" query to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger to obtain a private key $D_{\mathtt{ID}}$ associated with ID. (On receiving ID, the Challenger runs the private key extraction algorithm of IDTHDBM taking ID as input and returns a private key $D_{\mathtt{ID}}$ associated with ID to $\mathsf{A}^{\mathsf{CCA}}$.)
        · Get $D_{\mathtt{ID}}$ from the communication between the Challenger and $\mathsf{A}^{\mathsf{CCA}}$ and split it into $D_{\mathtt{ID},sem}$ and $D_{\mathtt{ID},user}$ using the (2, 2) secret-sharing technique presented in Section 5.5.1.
        · Return $D_{\mathtt{ID},user}$ to $\mathsf{A}^{\mathsf{CCA}\prime}$ as an answer.
        · Add $\langle \mathtt{ID}, D_{\mathtt{ID},user} \rangle$ to UserKeyList. Also, add $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ to SEMKeyList which consists of ⟨identity, corresponding SEM part of private key⟩ pairs.

We answer $\mathsf{A}^{\mathsf{CCA}\prime}$'s "SEM key extraction" queries in a similar way as we do for the "User key extraction" queries. Note that in this case, SEMKeyList and UserKeyList are also updated *concurrently*.

On receiving $\mathsf{A}^{\mathsf{CCA}\prime}$'s "SEM oracle" queries, each of which consists of $(\mathtt{ID}, C)$ where $C = (U, V, W)$, we perform the following:

– Search SEMKeyList for an entry $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$.
    * If there exists such an entry, extract $D_{\mathtt{ID},sem}$ from it. Then, check if $\hat{e}(P, W) = \hat{e}(U, H_3)$ for $H_3 = \mathsf{H}_3(U, V)$.
        · If C has passed this test, compute $\kappa_{sem} = \hat{e}(D_{\mathtt{ID},sem}, U)$ and return $\delta_{\mathtt{ID},sem,C} = (sem, \kappa_{sem})$ to $\mathsf{A}^{\mathsf{CCA}\prime}$.
        · Otherwise, return $\delta_{\mathtt{ID},sem,C} = (sem,$ "*Invalid Ciphertext*") to $\mathsf{A}^{\mathsf{CCA}\prime}$.
    * If $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ does not exist in SEMKeyList, do the following:
        · Forward ID as a "private key extraction" query to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger to obtain a private key $D_{\mathtt{ID}}$ associated with ID.
        · Get $D_{\mathtt{ID}}$ from the communication between the Challenger and $\mathsf{A}^{\mathsf{CCA}}$ and split it into $D_{\mathtt{ID},sem}$ and $D_{\mathtt{ID},user}$ using the (2, 2) secret-sharing technique.
        · Compute $\kappa_{sem} = \hat{e}(D_{\mathtt{ID},sem}, U)$ and return $\delta_{\mathtt{ID},sem,C} = (sem, \kappa_{sem})$ to $\mathsf{A}^{\mathsf{CCA}\prime}$.

· Add $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ and $\langle \mathtt{ID}, D_{\mathtt{ID},user} \rangle$ to SEMKeyList and UserKeyList respectively.

On receiving $\mathsf{A}^{\mathsf{CCA}\prime}$'s "User oracle" queries, each of which consists of $(\mathtt{ID}, C)$, we perform the following:

– Search SEMKeyList for an entry $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$.
  * If there exists such an entry, search UserKeyList for the corresponding entry $\langle \mathtt{ID}, D_{\mathtt{ID},user} \rangle$. (Recall that SEMKeyList and UserKeyList are updated concurrently. Hence if there exists an entry in SEMKeyList, we can always find the corresponding entry in UserKeyList.) Then, check if $\hat{e}(P, W) = \hat{e}(U, H_3)$ for $H_3 = \mathsf{H}_3(U, V)$.
    · If C has passed this test, compute $\kappa_{sem} = \hat{e}(D_{\mathtt{ID},sem}, U)$ and $\kappa_{user} = \hat{e}(D_{\mathtt{ID},user}, U)$, and combine them using the Lagrange interpolation technique and return the resulting value to $\mathsf{A}^{\mathsf{CCA}\prime}$.
    · Otherwise, return "*Reject*" to $\mathsf{A}^{\mathsf{CCA}\prime}$.
  * If $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ does not exist in SEMKeyList, do the following:
    · Forward ID as a "private key extraction" query to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger to obtain a private key $D_{\mathtt{ID}}$ associated with ID.
    · Get $D_{\mathtt{ID}}$ from the communication between the Challenger and $\mathsf{A}^{\mathsf{CCA}}$ and split it into $D_{\mathtt{ID},sem}$ and $D_{\mathtt{ID},user}$ using the (2, 2) secret-sharing technique.
    · Perform the same routine as we do for the case when $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ exists in SEMKeyList and return a special symbol "*Reject*" or a certain value to $\mathsf{A}^{\mathsf{CCA}\prime}$
    · Add $\langle \mathtt{ID}, D_{\mathtt{ID},sem} \rangle$ and $\langle \mathtt{ID}, D_{\mathtt{ID},user} \rangle$ to SEMKeyList and UserKeyList respectively.

In Phase 3, if $\mathsf{A}^{\mathsf{CCA}\prime}$ issues two equal-length plaintexts (M_0, M_1) and a target identity $\mathtt{ID}^*$, we forward $(M_0, M_1, \mathtt{ID}^*)$ to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger. On receiving $(M_0, M_1, \mathtt{ID}^*)$, the Challenger runs the private key extraction algorithm of IDTHDBM to get a private key $D_{\mathtt{ID}^*}$ associated with $\mathtt{ID}^*$ and runs the private key distribution algorithm of IDTHDBM to split $D_{\mathtt{ID}^*}$ into S_1^* and S_2^*. The Challenger returns S_2^* to $\mathsf{A}^{\mathsf{CCA}}$ as a corrupted party's private key. We then rename S_1^* and S_2^* as $D_{\mathtt{ID}^*,sem}$ and $D_{\mathtt{ID}^*,user}$ respectively and send $D_{\mathtt{ID}^*,user}$ to $\mathsf{A}^{\mathsf{CCA}\prime}$. (That is, the *strong* attacker $\mathsf{A}^{\mathsf{CCA}\prime}$ possesses the user part of private key.) Now, the Challenger chooses $\beta \in \{0, 1\}$ at random and runs the encryption algorithm E of IDTHDBM taking $(M_\beta, \mathtt{ID}^*)$ as input and gets a target ciphertext C^*. If the Challenger returns C^*, we send that to $\mathsf{A}^{\mathsf{CCA}\prime}$.

On receiving $\mathsf{A}^{\mathsf{CCA}\prime}$'s "User key extraction" and "SEM key extraction" queries in Phase 4, we answer them in the same way we did in Phase 2.

If $\mathsf{A}^{\mathsf{CCA}\prime}$ issues "SEM oracle" queries, each of which consists of $(\mathtt{ID}, C) \neq (\mathtt{ID}^*, C^*)$, in Phase 4, we perform the following:

- If $\mathtt{ID} \neq \mathtt{ID}^*$, answer the query in the same way we did for the SEM oracle query in Phase 2.
- If $\mathtt{ID} = \mathtt{ID}^*$ (in this case, $C \neq C^*$), do the following:
  - Forward C to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger as a "decryption share generation" query. (On receiving C, the Challenger runs the decryption share generation algorithm of IDTHDBM taking $(S_1^*(= D_{\mathtt{ID}^*,sem}), C)$ as input, gets a corresponding decryption share $\delta_{1,C}$ and returns it to $\mathsf{A}^{\mathsf{CCA}}$.)
  - Get $\delta_{1,C}$ from the communication between the Challenger and $\mathsf{A}^{\mathsf{CCA}}$. Then, do the following:
    If $\delta_{1,C} \neq (1, \text{"}\textit{Invalid Ciphertext}\text{"})$,
    - Take κ_1 out from $\delta_{1,C}$
    - Rename κ_1 as κ_{sem} and send $\delta_{\mathtt{ID}^*,sem,C} = (sem, \kappa_{sem})$ to $\mathsf{A}^{\mathsf{CCA}\prime}$.

    If $\delta_{1,C} = (1, \text{"}\textit{Invalid Ciphertext}\text{"})$,
    - Send $\delta_{\mathtt{ID}^*,sem,C} = (sem, \text{"}\textit{Invalid Ciphertext}\text{"})$ to $\mathsf{A}^{\mathsf{CCA}\prime}$.

If $\mathsf{A}^{\mathsf{CCA}\prime}$ issues "User oracle" queries, each of which consists of $(\mathtt{ID}, C) \neq (\mathtt{ID}^*, C^*)$, in Phase 4, we perform the following:

- If $\mathtt{ID} \neq \mathtt{ID}^*$, answer the query in the exactly same way we did for the "User oracle" query in Phase 2.
- If $\mathtt{ID} = \mathtt{ID}^*$ (in this case, $C \neq C^*$), do the following:
  - Forward C to $\mathsf{A}^{\mathsf{CCA}}$'s Challenger as a "decryption share generation" query. (On receiving C, the Challenger runs the decryption share generation algorithm of IDTHDBM taking $(S_1^*(= D_{\mathtt{ID}^*,sem}), C)$ as input, gets a corresponding decryption share $\delta_{1,C}$ and returns it to $\mathsf{A}^{\mathsf{CCA}}$.)
  - Get $\delta_{1,C}$ from the communication between the Challenger and $\mathsf{A}^{\mathsf{CCA}}$. Then, do the following:
    If $\delta_{1,C} \neq (1, \text{"}\textit{Invalid Ciphertext}\text{"})$,
    - Take out κ_1 from $\delta_{1,C}$, rename it as κ_{sem}, and form $\delta_{\mathtt{ID}^*,sem,C} = (sem, \kappa_{sem})$.

- Compute $\kappa_{user} = \hat{e}(D_{\mathtt{ID}^*,user}, U)$. (Recall that the $\mathsf{A}^{\mathsf{CCA}}$'s Challenger returned $D_{\mathtt{ID}^*,user}$ as a corrupted party's private key.)
- Check the validity of C. If C is invalid, that is, $\hat{e}(P, W) \neq \hat{e}(U, H_3)$ for $H_3 = \mathsf{H}_3(U, V)$, return "*Reject*". Otherwise, combine the shares κ_{user} and κ_{sem} using the Lagrange interpolation technique and return the resulting value to $\mathsf{A}^{\mathsf{CCA}\prime}$.

If $\delta_{1,C} = (1, \text{"}\textit{Invalid Ciphertext}\text{"})$,

- Send "*Reject*" to $\mathsf{A}^{\mathsf{CCA}\prime}$.

Finally, once $\mathsf{A}^{\mathsf{CCA}\prime}$ outputs a guess $\beta' \in \{0, 1\}$, we return it as $\mathsf{A}^{\mathsf{CCA}}$'s guess.

Note from the simulation that $\mathsf{A}^{\mathsf{CCA}\prime}$'s view in the real attack game is identical to it's view in Game $\mathbf{G_1}$. Note also that the bit β is uniformly chosen. Hence we have

$$\Pr[E_1] - \frac{1}{2} \geq \Pr[E_0] - \frac{1}{2}.$$

By definition of $\Pr[E_0]$ and $\Pr[E_1]$, we obtain

$$\mathbf{Succ}^{\mathrm{IND-IDTHD-CCA}}_{\mathtt{IDTHDBM},\mathsf{A}^{\mathsf{CCA}}}(k) \geq \mathbf{Succ}^{\mathrm{IND-mID-sCCA}}_{\mathtt{MIDEBM},\mathsf{A}^{\mathsf{CCA}\prime}}(k).$$

Considering the running time and the number of queries, we obtain the bound in the theorem statement. □

5.7 Brief Summary of the Results

In this chapter, we discussed issues related to the realization of ID-based threshold decryption and argued that it is important in practice to share the private key issued by the PKG rather than the master key of the PKG. We then proposed an ID-based threshold decryption scheme which satisfies such property and is provably secure against chosen-ciphertext attack. We also showed how our ID-based threshold decryption scheme can result in a mediated ID-based encryption scheme secure against "inside attack", whereby an attacker who possesses a user part of private key conducts chosen-ciphertext attack.

Chapter 6

Identity-Based Threshold Signature from the Bilinear Map

6.1 Introduction

6.1.1 Motivation

While threshold decryption is used to decentralize the power to decrypt, threshold signature is a useful tool for sharing the power to sign. And the idea of identity (ID)-based signature, presented by Shamir [108], is to create a signature on a message in such a way that any user can verify the signature using the signer's identifier information such as email address instead of the public key (a "number") embedded in a digital certificate. Combining these two concepts to realize "ID-based threshold signature" is the focus of this chapter.

One possible application of the ID-based threshold signature scheme can be considered in the following situation. Suppose that Alice, as a president of some company, has created an identity that represents the company and has a private key associated with the identity. Using the private key, she is able to sign any documents. However, concerning about the situation where she is away, she wants to delegate this power to a number of signature-generation servers so that a signature on a given message is jointly generated by those servers and any user can successfully verify the signature using the company's published identity if, and only if, the user obtains a certain number of partial signatures from the signature-generation servers.

6.1.2 Related Work

Threshold Signature in the Non-ID-Based Setting. In the non-ID-based setting, research on threshold signature schemes has been mainly focused on how to share the RSA [103] and ElGamal [49] signature functions. Even though there are a large number of research works related to this topic, we briefly review those most related to our research.

Cerecedo, Matsumoto and Imai's work [33] can be regarded as a first formal treatment of discrete-logarithm based threshold signatures appeared in the literature. They gave formal definition of unforgeability of threshold signature against chosen message attack and presented concrete schemes based on the ElGamal signature scheme and the Digital Signature Standard (DSS) [86]. Although it was later pointed out [59] that a stronger assumption than the unforgeability of the DSS scheme is needed for their threshold DSS signature scheme to be proven secure, their work contains some useful ideas worth being revived.

A complete solution to the construction of a threshold DSS signature scheme was given by Gennaro, Jarecki, Krawczyk, and Rabin [59]. They designed various distributed verifiable secret sharing schemes as building blocks to construct a robust and secure threshold DSS signature scheme. As exemplified in Stinson and Strobl's [120] threshold signature scheme based on the Schnorr signature scheme [104], their design technique can be applied to other discrete-logarithm based threshold signature signature schemes and also affected our scheme. Another notable application of this technique is Boldyreva's [24] threshold signature scheme based the Gap Diffie-Hellman (GDH) group in which the DDH problem (defined in Chapter 3) becomes computationally tractable while the CDH problem (defined in Chapter 3) remains intractable.

ID-Based Signature Scheme from the Bilinear Maps. Recently, several ID-based signature schemes based on the bilinear maps (pairings) have been proposed. These include Cha and Cheon's [34] scheme and Hess' [66] scheme. They gave a formal definition of unforgeability of ID-based signature against chosen message attack and proved their schemes are secure in this sense in the random oracle model [20] assuming the Computational Diffie-Hellman (CDH) problem is computationally intractable.

6.1.3 Contributions of This Chapter

To our knowledge, ID-based threshold signature has not been treated in the literature. In this chapter, we present a formal security notion for ID-based

threshold signature and give a concrete scheme whose security is based on that of Hess' ID-based signatures scheme [66] and hence the CDH problem.

Interestingly, the security of one of the verifiable secret-sharing schemes that we design to build our ID-based threshold signature scheme is relative to a slight variant of the Generalized Tate Inversion (GTI) problem that Joux [70] questioned how it can be used to construct a cryptographic protocol. Hence, our verifiable secret-sharing scheme gives a partial answer to Joux's question.

6.2 Preliminaries

6.2.1 Security Notion for ID-Based Signature

We first review the formal definition [34, 66] of a generic ID-based signature scheme, which we denote by "$\mathcal{IDS}$".

Definition 20 (ID-Based Signature) The ID-based signature scheme $\mathcal{IDS}$ consists of the following algorithms.

- A randomized key/common parameter generation algorithm $\mathsf{GC}(k)$: Given a security parameter $k \in \mathbb{N}$, this algorithm generates the PKG's master/public key pair $(sk_{\mathrm{PKG}}, pk_{\mathrm{PKG}})$. Then, it generates necessary common parameters, e.g., descriptions of hash functions and mathematical groups. The output of this algorithm denoted by cp includes such parameters and the PKG's public key pk_{PKG}. Note that cp is given to all interested parties while the matching master key sk_{PKG} is kept secret.
- A deterministic private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: Given an identity $\mathtt{ID}$, this algorithm generates a private key associated with $\mathtt{ID}$, denoted by $sk_{\mathtt{ID}}$.
- A randomized signature generation algorithm $\mathsf{S}(cp, sk_{\mathtt{ID}}, M)$: Given a private key $sk_{\mathtt{ID}}$ associated with $\mathtt{ID}$ and a message M, this algorithm generates a signature σ on M.
- A deterministic signature verification algorithm $\mathsf{V}(cp, \mathtt{ID}, M, \sigma)$: Given a signer's identity $\mathtt{ID}$, a message M and its signature σ, this algorithm returns "*Valid*" if σ is valid. Otherwise, it returns "*Invalid*".

We now review the unforgeability notion for the $\mathcal{IDS}$ scheme against chosen message attack [34, 66], which we denote by "UF-IDS-CMA".

Definition 21 (UF-IDS-CMA) Let $\mathsf{A}^{\mathsf{CMA}}$ be an attacker assumed to be a probabilistic Turing machine taking a security parameter k as input. Consider the following game in which $\mathsf{A}^{\mathsf{CMA}}$ interacts with the "Challenger".

> **Phase 1**: The Challenger runs the PKG's key/common parameter generation algorithm GC of $\mathcal{IDS}$ taking a security parameter k as input. The Challenger gives $\mathsf{A}^{\mathsf{CMA}}$ the resulting common parameter cp which includes the PKG's public key pk_{PKG}. However, the Challenger keeps the master key sk_{PKG} secret from $\mathsf{A}^{\mathsf{CMA}}$.
>
> **Phase 2**: $\mathsf{A}^{\mathsf{CMA}}$ issues a number of private key extraction queries, each of which consists of $\mathtt{ID}$. On receiving $\mathtt{ID}$, the Challenger runs the private key extraction algorithm EX of $\mathcal{IDS}$ taking $\mathtt{ID}$ as input and obtains a corresponding private key $sk_{\mathtt{ID}}$. Then, the Challenger gives $sk_{\mathtt{ID}}$ to $\mathsf{A}^{\mathsf{CMA}}$. In addition to the key extraction queries, $\mathsf{A}^{\mathsf{CMA}}$ issues a number of signature generation queries, each of which consists of $(\mathtt{ID}, M)$. On receiving $(\mathtt{ID}, M)$, the Challenger first runs the private key extraction algorithm EX of $\mathcal{IDS}$ taking $\mathtt{ID}$ as input and obtains a corresponding private key $sk_{\mathtt{ID}}$. The Challenger then runs the signature generation algorithm S of $\mathcal{IDS}$ taking $sk_{\mathtt{ID}}$ as input and gives a resulting signature σ to $\mathsf{A}^{\mathsf{CMA}}$.
>
> **Phase 3**: $\mathsf{A}^{\mathsf{CMA}}$ outputs $(\tilde{\mathtt{ID}}, \tilde{M}, \tilde{\sigma})$, where $\tilde{\sigma}$ is a valid signature of a message $\tilde{M}$ and $\tilde{\mathtt{ID}}$ is a corresponding identity. A restriction here is that $\mathsf{A}^{\mathsf{CMA}}$ must not make a private key extraction query for $\tilde{\mathtt{ID}}$, and it must not make a signature generation query for $(\tilde{\mathtt{ID}}, \tilde{M})$. (Note that $\mathsf{A}^{\mathsf{CMA}}$ can make a signature generation query for $(\tilde{\mathtt{ID}}, M'(\neq \tilde{M}))$ and $(\mathtt{ID}'(\neq \tilde{\mathtt{ID}}), \tilde{M})$.)

We define $\mathsf{A}^{\mathsf{CMA}}$'s success by

$$\mathbf{Succ}^{\mathrm{UF-IDS-CMA}}_{\mathcal{IDS},\mathsf{A}^{\mathsf{CMA}}}(k) = \Pr[\mathsf{V}(cp, \tilde{\mathtt{ID}}, \tilde{M}, \tilde{\sigma}) = 1].$$

We denote by $\mathbf{Succ}^{\mathrm{UF-IDS-CMA}}_{\mathcal{IDS}}(t_{CMA}, q_E, q_S)$ the maximum of the attacker $\mathsf{A}^{\mathsf{CMA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CMA}}$ having running time t_{CMA} and making at most q_E key extraction queries and q_S signature generation queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The scheme $\mathcal{IDS}$ is said to be UF-IDS-CMA secure if $\mathbf{Succ}^{\mathrm{UF-IDS-CMA}}_{\mathcal{IDS}}(t_{CMA}, q_E, q_S)$ is negligible in k.

6.2.2 Computational Primitives

To keep this chapter self-contained, we briefly review the basic facts about the admissible bilinear map. We then present related computational problems on which the various secret-sharing and our ID-based threshold signature schemes are based.

Bilinear Map

The admissible bilinear map $\hat{e}$, which we will simply call a "Bilinear map", is defined over two groups of the same prime order q denoted by $\mathcal{G}$ and $\mathcal{F}$ in which the Computational Diffie-Hellman problem is hard. We will use an additive notation to describe the operation in $\mathcal{G}$ while we will use a multiplicative notation for the operation in $\mathcal{F}$. In practice, the group $\mathcal{G}$ is implemented using a group of points on certain elliptic curves, each of which has a small MOV exponent [81] while the group $\mathcal{F}$ will be implemented using a subgroup of the multiplicative group of a finite field. The Bilinear pairing $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$ has the following properties [27].

- Bilinear: $\hat{e}(aR_1, bR_2) = \hat{e}(R_1, R_2)^{ab}$, where $R_1, R_2 \in \mathcal{G}$ and $a, b \in \mathbb{Z}_q^*$.
- Non-degenerate: $\hat{e}$ does not send all pairs of points in $\mathcal{G} \times \mathcal{G}$ to the identity in $\mathcal{F}$. (Hence, if R is a generator of $\mathcal{G}$ then $\hat{e}(R, R)$ is a generator of $\mathcal{F}$.)
- Computable: For all $R_1, R_2 \in \mathcal{G}$, the pairing $\hat{e}(R_1, R_2)$ is efficiently computable.

Computational Diffie-Hellman Problem

We now the CDH problem in the group $\mathcal{G}$. Note that throughout this chapter, the groups $\mathcal{G}$ and $\mathcal{F}$ are as defined in the above description of the Bilinear map.

Definition 22 (CDH) Let $\mathcal{G}$ be a group of a prime order $q \geq 2^k$, where k is security parameter. Let $P \in \mathcal{G}^*$ be a generator of $\mathcal{G}$. Let $\mathsf{A}^{\mathsf{CDH}}$ be an attacker modelled as a probabilistic Turing machine taking the security parameter k as input. Suppose that a and b are uniformly chosen at random from $\mathbb{Z}_q^*$ and aP and bP are computed.

$\mathsf{A}^{\mathsf{CDH}}$ is to solve the following problem:

- Given $(\mathcal{G}, q, P, aP, bP)$ for random $a, b \in \mathbb{Z}_q^*$, compute the Diffie-Hellman key abP of aP and bP.

We define $\mathsf{A}^{\mathsf{CDH}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{CDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{CDH}}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{CDH}} \text{ outputs } abP].$$

We denote by $\mathbf{Succ}^{\mathrm{CDH}}_{\mathcal{G}}(t_{\mathsf{CDH}})$ the maximal success probability $\mathbf{Succ}^{\mathrm{CDH}}_{\mathcal{G},\mathsf{A}^{\mathsf{CDH}}}(k)$ over all attackers having running time bounded by t_{CDH} which is polynomial in the security parameter k.

The CDH problem is said to be intractable if $\mathbf{Succ}^{\mathrm{CDH}}_{\mathcal{G}}(t_{CDH})$ is negligible in k.

It is believed that the CDH problem in the group $\mathcal{G}$ is intractable. On the contrary, the Decisional Diffe-Hellman (DDH) problem in this group can be solved in polynomial time with the help of the Bilinear map: Given (P, aP, bP, cP) where $a, b, c \in \mathbb{Z}_q^*$, one can decide whether $c = ab$ by checking whether $\hat{e}(P, cP) = \hat{e}(aP, bP)$. For this reason, the group $\mathcal{G}$ is call the Gap Diffie-Hellman (GDH) group.

Modified Generalized Bilinear Inversion Problem

Recently, Joux [70] proposed a new computational problem related to the Tate pairing, called the Generalized Tate Inversion (GTI) problem. Informally, the GTI problem refers to the computational problem in which an attacker, given $h \in \mathcal{F}$, is to find a pair $(S, T) \in \mathcal{G} \times \mathcal{G}$ such that $e(S, T) = h$, where e denotes the Tate pairing.

We modify the above GTI problem and obtain a new computational problem, which we call "modified Generalized Bilinear Inversion (mGBI)". Note that in the mGBI problem, the admissible bilinear map ("Bilinear map") replaces the role of the Tate pairing in the GTI problem.

Definition 23 (mGBI) Let $\mathcal{G}$ and $\mathcal{F}$ be two groups of a prime order $q \geq 2^k$, where k is security parameter. Let $P \in \mathcal{G}^*$ be a generator of $\mathcal{G}$. Suppose that there exists a bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \to \mathcal{F}$. Let $\mathsf{A}^{\mathsf{mGBI}}$ be an attacker modelled as a probabilistic Turing machine taking the security parameter k as input. Suppose that h is randomly chosen from $\mathcal{F}$.

$\mathsf{A}^{\mathsf{mGBI}}$ is to solve the following problem:

- Given $h \in \mathcal{F}$ and $P \in \mathcal{G}$, compute $S \in \mathcal{G}$ such that $\hat{e}(S, P) = h$.

We define $\mathsf{A}^{\mathsf{mGBI}}$'s success probability by

$$\mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G},\mathsf{A}^{\mathsf{mGBI}}}(k) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{A}^{\mathsf{mGBI}}(\mathcal{G}, q, P, h) = S].$$

We denote by $\mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G}}(t_{\mathsf{mGBI}})$ the maximal success probability $\mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G},\mathsf{A}^{\mathsf{mGBI}}}(k)$ over all attackers having running time bounded by t_{mGBI} which is polynomial in the security parameter k.

The mGBI problem is said to be intractable if $\mathbf{Succ}^{\mathrm{mGBI}}_{\mathcal{G}}(t_{mGBI})$ is negligible in k.

In fact, the mGBI assumption (that is, the mGBI problem is intractable) is weaker than the GBI (Generalized Bilinear Inversion) assumption which is naturally derived from the GTI assumption by replacing the Tate pairing by the admissible bilinear map, as sketched below: Assume that an attacker $\mathsf{A}^{\mathsf{GBI}}$ for the GBI problem has access to $(\mathcal{G}, \hat{e}, h)$, where $h \in \mathcal{F}$. First, $\mathsf{A}^{\mathsf{GBI}}$ chooses a generator P for $\mathcal{G}$. It then runs an attacker $\mathsf{A}^{\mathsf{mGBI}}$ for the mGBI problem providing $(\mathcal{G}, \hat{e}, h, P)$ as input. If $\mathsf{A}^{\mathsf{mGBI}}$ outputs S, $\mathsf{A}^{\mathsf{GBI}}$ sets $T = P$ and returns (S, T).

6.3 Security Notions of ID-Based Threshold Signature

6.3.1 Security Notion for ID-Based Threshold Signature

High Level Description of ID-based Threshold Signature

We first present a high level description of a (generic) (t, n) ID-based threshold signature scheme, which we call "$\mathcal{IDTHS}$". The $\mathcal{IDTHS}$ scheme consists of algorithms GC, EX, DK, S, and V

The key/common parameter generation algorithm GC is run by the trusted PKG to generate its master/public key pair and all the necessary common parameters. The PKG's public key and the common parameters are given to every interested party.

On receiving a user's private key extraction query which consists of an identity, the PKG runs the private key extraction algorithm EX to generate a private key associated with the queried identity.

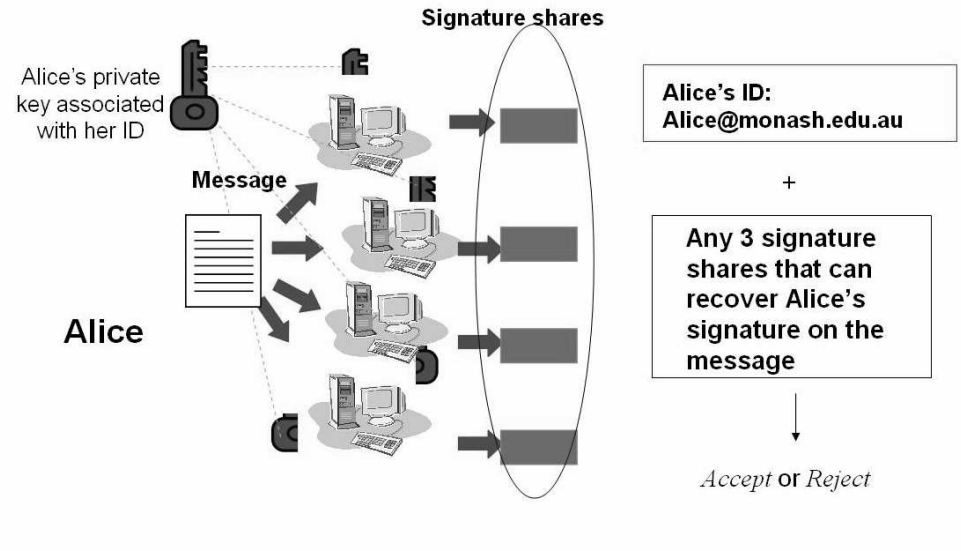


Figure 6.1: (3,4)-Identity-Based Threshold Signature

Like our ID-based threshold decryption scheme presented in the previous chapter, an authorized dealer who possesses the private key (associated with an identity) runs the private key distribution algorithm DK to distribute the private key into n signature generation servers. DK makes use of a proper secret-sharing technique to generate shares of the private key as well as verification keys that will be used for checking the validity of partial signature shares. Each share of the private key and its corresponding verification key are sent to an appropriate signature generation server. Having obtained the shares of the private key and the matching verification keys, the signature generation servers keep their private key shares secret but publicize the verification keys. Note here that the entity that runs DK can be either a normal user (such as Alice in the example given in Section 6.1.1) or the PKG, depending on the cryptographic services that the PKG can offer. – If the PKG is capable of organizing threshold signature, the PKG can run DK, but if the PKG has the only functionality of issuing private keys for users, the entity running DK would be a trusted normal user. Note also that like the ID-based threshold decryption scheme presented in Chapter 5, a private key associated with an identity is shared in our ID-based threshold signature scheme. As discussed in the previous chapter, this approach is more practical than Boneh and Franklin's [27] approach that distributes the master key of the PKG into a number of other PKGs (called the "Distributed PKGs") to perform threshold decryption or threshold signature generation, since the latter

approach requires the distributed PKGs to be involved on-line in performing such cryptographic operations. Obviously, this creates a bottleneck on the PKGs and also violates one of the basic requirements of an ID-based encryption scheme, "the PKG can be closed after key generation", which was envisioned by Shamir in his original proposal of ID-based cryptography [108]. Moreover, it causes a scalability problem when the number of available distributed PKGs is not matched against the number of decryption servers required, say, there are only 3 available PKGs while a certain application requires 5 signature generation servers.

Given a set of common parameters generated by GC, a share of the private key associated with an identity, and a message, n signature generation servers jointly generate a signature for a given message by the signature generation algorithm S.

When a user collects valid signature shares from at least t servers, the whole signature on the given message can be reconstructed and the validity of this signature can be checked by running the signature verification algorithm V.

Figure 6.1 illustrates the scenario of $(3, 4)$-ID-based threshold signature. Below, we formally define $\mathcal{IDTHS}$.

Definition 24 (ID-Based Threshold Signature) A (t, n) ID-based threshold signature scheme $\mathcal{IDTHS}$ consists of the following algorithms.

- A randomized key/common parameter generation algorithm $\mathsf{GC}(k)$: Given a security parameter $k \in \mathbb{N}$, this algorithm generates the PKG's master/public key pair $(sk_{\mathrm{PKG}}, pk_{\mathrm{PKG}})$. It then generates necessary common parameters, e.g., descriptions of hash functions and mathematical groups. The output of this algorithm denoted by cp includes such parameters and the PKG's public key pk_{PKG}. Note that cp is given to all interested parties while the matching master key sk_{PKG} is kept secret.

- A deterministic private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: Given an identity $\mathtt{ID}$, this algorithm generates a private key associated with $\mathtt{ID}$, denoted by $sk_{\mathtt{ID}}$.

- A randomized private key distribution algorithm $\mathsf{DK}(cp, sk_{\mathtt{ID}}, n, t)$: Given a private key $sk_{\mathtt{ID}}$ associated with an identity $\mathtt{ID}$, a number of signature generation servers n and a threshold parameter t, this algorithm generates n shares of $sk_{\mathtt{ID}}$ and provides each one to the signature generation servers $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$. It also generates a set of verification keys that can be used to check the validity of each shared private key. We denote the shared private

keys and the matching verification keys by $\{sk_i\}_{i=1,\ldots,n}$ and $\{vk_i\}_{i=1,\ldots,n}$, respectively. Note that each (sk_i, vk_i) is sent to the signature generation server Γ_i, then Γ_i publishes vk_i but keeps sk_i as secret.

- A randomized signature generation algorithm $\mathsf{S}(cp, sk^i_{\mathtt{ID}}, M)$: Given the common parameter cp generated by GC, a share $sk^i_{\mathtt{ID}}$ of the private key $sk_{\mathtt{ID}}$ associated with $\mathtt{ID}$ and a message M, n signature generation servers jointly generate a signature σ for the message M. Note that partial signatures of M computed by each server may be broadcast during the execution of S.

- A deterministic signature verification algorithm $\mathsf{V}(cp, \mathtt{ID}, M, \sigma)$: Given a signer's identity $\mathtt{ID}$, a message M and its signature σ, this algorithm checks the validity of σ. The output of this algorithm is either "*Valid*" or "*Invalid*".

Unforgeability for ID-Based Threshold Signature against Chosen Message Attack

We now present an unforgeability notion for $\mathcal{IDTHS}$ against chosen message attack, which we call "UF-IDTHS-CMA".

Definition 25 (UF-IDTHS-CMA) Let $\mathsf{B}^{\mathsf{CMA}}$ be an attacker assumed to be a probabilistic Turing machine taking a security parameter k as input. Consider the following game in which $\mathsf{B}^{\mathsf{CMA}}$ interacts with the "Challenger".

Phase 1: $\mathsf{B}^{\mathsf{CMA}}$ corrupts $t-1$ signature generation servers. (That is, the attacker is assumed to be static [59, 60].)

Phase 2: $\mathsf{B}^{\mathsf{CMA}}$ issues a number of private key extraction queries, each of which consists of $\mathtt{ID}$. On receiving $\mathtt{ID}$, the Challenger runs the key extraction algorithm EX of $\mathcal{IDTHS}$ taking $\mathtt{ID}$ as input and obtains a corresponding private key $sk_{\mathtt{ID}}$. The Challenger gives $sk_{\mathtt{ID}}$ to $\mathsf{B}^{\mathsf{CMA}}$.

Phase 3: $\mathsf{B}^{\mathsf{CMA}}$ submits a target identity $\mathtt{ID}^*$. On receiving $\mathtt{ID}^*$, the Challenger runs the key extraction algorithm EX of $\mathcal{IDTHS}$ taking $\mathtt{ID}$ as input and obtains a corresponding private key $sk_{\mathtt{ID}^*}$. Subsequently, it runs the private key distribution algorithm DK of $\mathcal{IDTHS}$ taking $sk_{\mathtt{ID}^*}$ as input to share it among n signature generation servers. We denote the shared keys by $sk^i_{\mathtt{ID}^*}$ for $i = 1, \ldots, n$. The Challenger gives $sk^i_{\mathtt{ID}^*}$ for $i = 1, \ldots, t-1$ (private keys for the corrupted servers) to $\mathsf{B}^{\mathsf{CMA}}$.

Phase 4: $\mathsf{B}^{\mathsf{CMA}}$ issues a number of signature generation queries, each of which consists of a message denoted by M. On receiving M, the Challenger, on behalf of the uncorrupted servers, runs the signature generation algorithm S of $\mathcal{IDTHS}$ taking $sk^i_{\mathtt{ID}}$ for $i = t, \ldots, n$ and M as input, and responds to $\mathsf{B}^{\mathsf{CMA}}$ with σ output by the signature generation algorithm S of $\mathcal{IDTHS}$. Note that in this phase, $\mathsf{B}^{\mathsf{CMA}}$ is allowed to issue private key extraction queries (identities) except for $\mathtt{ID}^*$. Note also that $\mathsf{B}^{\mathsf{CMA}}$ is allowed to see partial signature broadcast during the execution of S.

Phase 5: $\mathsf{B}^{\mathsf{CMA}}$ outputs $(\mathtt{ID}^*, \tilde{M}, \tilde{\sigma})$, where $\tilde{\sigma}$ is a valid signature of a message $\tilde{M}$ and $\mathtt{ID}^*$ is a corresponding identity. A restriction here is that $\mathsf{B}^{\mathsf{CMA}}$ must not make a private key extraction query for $\mathtt{ID}^*$, and it must not make a signature generation query for $\tilde{M}$.

We define $\mathsf{B}^{\mathsf{CMA}}$'s success by

$$\mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS},\mathsf{B}^{\mathsf{CMA}}}(k) = \Pr[\mathsf{V}(cp, \mathtt{ID}^*, \tilde{M}, \tilde{\sigma}) = 1].$$

We denote by $\mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS}}(t_{CMA}, q_E, q_S)$ the maximum of the attacker $\mathsf{B}^{\mathsf{CMA}}$'s success over all attackers $\mathsf{A}^{\mathsf{CMA}}$ having running time t_{CMA} and making at most q_E key extraction queries and q_S signature generation queries. Note that the running time and the number of queries are all polynomial in the security parameter k.

The ID-based threshold signature scheme $\mathcal{IDTHS}$ is said to be UF-IDTHS-CMA secure if $\mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS}}(t_{CMA}, q_E, q_S)$ is negligible in k.

We now define robustness of the ID-based threshold signature scheme.

Definition 26 (Robustness) A (t, n) ID-based threshold signature scheme $\mathcal{IDTHS}$ is said to be *robust* if it computes a correct output even in the presence of a malicious attacker that makes the corrupted signature generation servers deviate from the normal execution.

6.3.2 Relationship between UF-IDS-CMA and UF-IDTHS-CMA

Motivated by Gennaro et al.'s [59] methodology for proving the security of threshold signature, we define simulatability of $\mathcal{IDTHS}$ as follows.

Definition 27 (Simulatability of $\mathcal{IDTHS}$) Let $\mathcal{IDTHS}$ = (GC, EX, DK, S, V) be a (t, n) ID-based threshold signature scheme. The scheme $\mathcal{IDTHS}$ is said to be *simulatable* if the following conditions hold.

1. DK is simulatable: There exists a simulator SIM_{DK} that, given a common parameter *cp* generated by GC of $\mathcal{IDTHS}$, and an identity ID, can simulate the view of the attacker on an execution of DK of $\mathcal{IDTHS}$.
2. S is simulatable: There exists a simulator SIM_{S} that, given a common parameter *cp* generated by GC of $\mathcal{IDTHS}$, an identity ID, a message M, and a signature σ on M, $t-1$ shares of the private key that matches to ID of the corrupted signature generation servers, and the public outputs of DK of $\mathcal{IDTHS}$, can simulate the view of the attacker on an execution of S of $\mathcal{IDTHS}$.

We now state and prove the following theorem regarding the relationship between the security of $\mathcal{IDTHS}$ and that of $\mathcal{IDS}$. The implication of the theorem is that if we have a UF-IDS-CMA secure ID-based signature scheme, we can use it as a building block to construct an UF-IDTHS-CMA secure ID-based threshold signature scheme by ensuring simulatability.

Theorem 7 *If the $\mathcal{IDTHS}$ scheme is simulatable and the $\mathcal{IDS}$ scheme which is associated with the $\mathcal{IDTHS}$ scheme is UF-IDS-CMA secure, then the $\mathcal{IDTHS}$ is UF-IDTHS-CMA secure. Concretely, we obtain the following bound:*

$$\mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS}}(t_{CMA}, q_E, q_S) \leq \mathbf{Succ}^{\mathrm{UF-IDS-CMA}}_{\mathcal{IDS}}(t'_{CMA}, q'_E, q'_S),$$

where $t'_{CMA} = t_{CMA} + T_{SIM_{\mathsf{DK}}} + T_{SIM_{\mathsf{S}}}$, $q'_E = q_E$, and $q'_S = 1$. Here, $T_{SIM_{\mathsf{DK}}}$ and $T_{SIM_{\mathsf{S}}}$ denote the running time of the simulators SIM_{DK} and SIM_{S} respectively.

Proof. Let $\mathsf{B}^{\mathsf{CMA}}$ denote an attacker that defeats the UF-IDTHS-CMA security of the $\mathcal{IDTHS}$ scheme. Let $\mathsf{A}^{\mathsf{CMA}}$ denote an attacker that defeats the UF-IDS-CMA security of the $\mathcal{IDS}$ scheme.

We show how the view of $\mathsf{B}^{\mathsf{CMA}}$ in the real attack game of UF-IDTHS-CMA (Definition 25), which we denote by $\mathbf{G_0}$, can be simulated to obtain a new game $\mathbf{G_1}$ which is related to the ability of the attacker $\mathsf{A}^{\mathsf{CMA}}$ to defeat the UF-IDS-CMA security of the $\mathcal{IDS}$ scheme, under the assumption that $\mathcal{IDTHS}$ is simulatable.

- Game $\mathbf{G_0}$: As mentioned, this game is identical to the real attack game described in Definition 25. We denote by E_0 the event that $\mathsf{B}^{\mathsf{CMA}}$ outputs a

valid message/signature pair as a forgery. We use a similar notation E_1 for Game $\mathbf{G_1}$. Since Game $\mathbf{G_0}$ is the same as the real attack game, we have

$$\Pr[E_0] = \mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS},\mathsf{B}^{\mathsf{CMA}}}(k).$$

- Game $\mathbf{G_1}$: First, we replace $\mathsf{B}^{\mathsf{CMA}}$'s common parameter by $\mathsf{A}^{\mathsf{CMA}}$'s common parameter cp. We then do the following.

  Whenever $\mathsf{B}^{\mathsf{CMA}}$ issues a private key extraction query $\mathtt{ID}$ in Phase 2 of the attack game in Definition 25, we intercept it and forward $\mathtt{ID}$ as $\mathsf{A}^{\mathsf{CMA}}$'s private key extraction query to $\mathsf{A}^{\mathsf{CMA}}$'s Challenger. On receiving $\mathtt{ID}$, the Challenger runs the key extraction algorithm EX of $\mathcal{IDS}$ taking $\mathtt{ID}$ as input and returns the resulting private key $sk_{\mathtt{ID}}$. We simply send $sk_{\mathtt{ID}}$ back to $\mathsf{B}^{\mathsf{CMA}}$.

  If $\mathsf{B}^{\mathsf{CMA}}$ submits a target identity $\mathtt{ID}^*$ in Phase 3, we run SIM_{DK} taking cp and $\mathtt{ID}^*$ as input to simulate the view of $\mathsf{B}^{\mathsf{CMA}}$. (Note that during the execution of SIM_{DK}, $t-1$ shares of the private key $sk_{\mathtt{ID}^*}$ of corrupted signature generation servers are output, which we send to $\mathsf{B}^{\mathsf{CMA}}$. Note also that we do not ask $\mathtt{ID}^*$ to $\mathsf{A}^{\mathsf{CMA}}$'s Challenger to get a corresponding private key and hence we do not know the value $sk_{\mathtt{ID}^*}$.)

  In Phase 4, if $\mathsf{B}^{\mathsf{CMA}}$ issues a signature generation query M, we intercept it and forward $(\mathtt{ID}^*, M)$ as $\mathsf{A}^{\mathsf{CMA}}$'s signature generation query to $\mathsf{A}^{\mathsf{CMA}}$'s Challenger to get a corresponding signature σ. Having obtained σ, we run SIM_{S} taking cp, the outputs generated by SIM_{DK}, which includes corrupted $t-1$ shares of the private key $sk_{\mathtt{ID}^*}$, the target identity $\mathtt{ID}^*$, and the message/signature pair (M, σ) as input. We then send SIM_{S}'s outputs to $\mathsf{B}^{\mathsf{CMA}}$. $\mathsf{B}^{\mathsf{CMA}}$'s private key extraction queries other than $\mathtt{ID}^*$ at this stage are dealt with in the same way as done in the simulation of Phase 2.

  If $\mathsf{B}^{\mathsf{CMA}}$ outputs $(\mathtt{ID}^*, \tilde{M}, \tilde{\sigma})$ in Phase 5, we intercept it and return it as $\mathsf{A}^{\mathsf{CMA}}$'s forgery. Note from the simulation that $\mathsf{B}^{\mathsf{CMA}}$'s view in the real attack game is identical to it's view in Game $\mathbf{G_1}$ as long as the $\mathcal{IDTHS}$ is simulatable. Hence we have

  $$\Pr[E_1] \geq \Pr[E_0].$$

  By definition of $\Pr[E_0]$ and $\Pr[E_1]$, we obtain

  $$\mathbf{Succ}^{\mathrm{UF-IDS-CMA}}_{\mathcal{IDS},\mathsf{A}^{\mathsf{CMA}}}(k) \geq \mathbf{Succ}^{\mathrm{UF-IDTHS-CMA}}_{\mathcal{IDTHS},\mathsf{B}^{\mathsf{CMA}}}(k).$$

Considering the running time and the number of queries, we obtain the bound in the theorem statement.

□

6.4 Building Blocks for Our ID-based Threshold Signature

6.4.1 Hess' ID-Based Signature Scheme

We first review Hess' ID-based signature scheme [66], which we denote by "HessIDS". We will use this as a base ID-based signature scheme to construct our ID-based threshold signature scheme. Note that the HessIDS scheme was proven to be unforgeable against chosen message attack in the random oracle model [20] assuming that the CDH problem is intractable.

- A key/common parameter generation algorithm $\mathsf{GC}(k)$: This algorithm is run by the PKG to generate its master/public key pair and all the necessary common parameters.
  - Choose a group $\mathcal{G}$ of prime order $q \geq 2^k$, whose generator is P. Specify the Bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$.
  - Pick a master key s uniformly at random from $\mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
  - Choose two hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathcal{G}$ and $\mathsf{H}_2 : \{0,1\}^* \times \mathcal{F} \rightarrow \mathbb{Z}_q^*$.
  - Keep s as secret and publish $(\mathcal{G}, q, P, P_{pub}, \hat{e}, \mathsf{H}_1, \mathsf{H}_2)$.
- A private key extraction algorithm $\mathsf{EK}(cp, \mathtt{ID})$: This algorithm is run by the PKG on receiving a private key extraction query from any user who wants to extract a private key that matches to an identity ID.
  - Given ID, compute $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = sQ_{\mathtt{ID}}$.
  - Output $D_{\mathtt{ID}}$.
- A signature generation algorithm $\mathsf{S}(cp, D_{\mathtt{ID}}, M)$: This algorithm is run by a user who wants to sign a message M using a private key $D_{\mathtt{ID}}$.

  - Choose k uniformly at random from $\mathbb{Z}_q^*$.
  - Compute $K = kP$ and $\gamma = \hat{e}(K, P)(= \hat{e}(P, P)^k)$.
  - Compute $v = \mathsf{H}_2(M, \gamma)$ and $U = vD_{\mathtt{ID}} + K$.
  - Output a signature $\sigma = (U, v)$.

- A signature verification algorithm $\mathsf{V}(cp, \mathtt{ID}, \sigma)$: This algorithm is run by any user who wants to verify a signature σ on message M using the signer's identity $\mathtt{ID}$.

  - Compute $\gamma = \hat{e}(U, P)\hat{e}(Q_{\mathtt{ID}}, -P_{pub})^v$, where $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$.
  - If $\mathsf{H}_1(m, \gamma) = v$ then output "*Accept*", otherwise output "*Reject*".

Note that in the security proof of our ID-based threshold signature scheme, we will need a value $\gamma = \hat{e}(K, P)$, which is not explicitly appeared in `HessIDS`. However, this can be easily derived from a signature (U, v) by computing $\gamma = \hat{e}(U, P)\hat{e}(Q_{\mathtt{ID}}, -P_{pub})^v$. Hence, whether it is appeared or not, γ does not affect the security.

6.4.2 Review of Secret-Sharing over $\mathcal{G}$

To construct an ID-based threshold signature scheme from the above `HessIDS` scheme, we need to distribute the values K and $D_{\mathtt{ID}}$. This can be done efficiently using the secret-sharing scheme over the group $\mathcal{G}$ presented in Section 5.5.1. To maintain this chapter self-contained, we review it in the following.

Distribution Phase: Let q be a prime order of a group $\mathcal{G}$ of points on some elliptic curve. Let $S \in \mathcal{G}^*$ be a secret (point) to share. Suppose that we have chosen integers t and n satisfying $1 \leq t \leq n < q$.

First, we pick $F_1, F_2, \ldots, F_{t-1}$ uniformly at random from $\mathcal{G}^*$. Then, we define a polynomial-like function $F : \mathbb{N} \cup \{0\} \to \mathcal{G}$, which we call "PLF" throughout this chapter, such that

$$F(x) = S + \sum_{l=1}^{t-1} x^l F_l \in \mathcal{G}.$$

Define $t - 1$ as a "degree".

Now, we compute $S_i = F(i) \in \mathcal{G}$ for $1 \leq i \leq n$ and send (i, S_i) to the i-th member of the group of cardinality n. Note that when $i = 0$, we obtain the secret itself, that is, $S = F(0)$.

Reconstruction Phase: Let $\Phi \subset \{1, \ldots, n\}$ b set such that $|\Phi| \geq t$, where $|\cdot|$ denotes the cardinality of a given set. The function $F(x)$ can be reconstructed by computing

$$F(x) = \sum_{j \in \Phi} \pi_{xj}^{\Phi} S_j \text{ where } \pi_{xj}^{\Phi} = \prod_{\iota \in \Phi, \iota \neq j} \frac{x - \iota}{j - \iota} \in \mathbb{Z}_q.$$

6.4.3 Computationally Secure Verifiable Secret-Sharing Scheme Based on the Bilinear Map

Verifiable Secret-Sharing (VSS) is a useful tool for protecting threshold signature schemes from malicious attackers that cause involving parties to divert from the predefined protocol. In other words, VSS gives threshold signature schemes robustness.

Since our ID-based threshold signature scheme is of discrete-logarithm type, the various discrete-logarithm type VSS schemes, e.g., [50, 93] can be considered as building blocks for our VSS schemes. However, we modify those schemes as the base secret-sharing scheme presented in the previous section has a different structure than Shamir's original secret-sharing scheme, and the Bilinear map should extensively be used in our ID-based threshold signature scheme.

Our first VSS scheme, which we call "Computationally secure Verifiable Secret-Sharing scheme based on the Bilinear Map (`CVSSBM`)", is motivated by Feldman's VSS scheme [50]. This scheme will be used to distribute a user's private key $D_{\mathtt{ID}}$ in the `HessIDS` scheme into a number of signature generation servers.

Description of `CVSSBM`

We describe the `CVSSBM` scheme as follows.

Let $(\mathcal{G}, q, P, \hat{e})$ be a set of parameters, as defined in Section 6.2.2. Suppose that a threshold t and the number of parties n satisfy $1 \leq t \leq n < q$. To share a secret $S \in \mathcal{G}^*$ out among n parties, a dealer performs the following:

1. Choose $F_1, \ldots, F_{t-1}$ uniformly at random from $\mathcal{G}^*$, construct a PLF $F(x) = S + xF_1 + \cdots + x^{t-1}F_{t-1} \in \mathcal{G}$ for $x \in \mathbb{N} \cup \{0\}$ and compute $S_i = F(i)$ for $i = 0, \ldots, n$. Set $S_0 = S$.

2. Send S_i to party Γ_i for $i = 1, \ldots, n$ secretly. Broadcast $\alpha_0 = \hat{e}(S, P)$ and $\alpha_j = \hat{e}(F_j, P)$ for $j = 1, \ldots, t-1$.

3. Each party Γ_i then checks whether its share S_i is valid by computing

$$\hat{e}(S_i, P) = \prod_{j=0}^{t-1} \alpha_j^{i^j}. \tag{6.1}$$

Analysis of `CVSSBM`

The `CVSSBM` scheme satisfies correctness. More precisely, we state and prove the following lemma.

Lemma 9 *In* `CVSSBM`*, shares held by all the uncorrupted parties can be interpolated to a unique PLF of degree* $t-1$*, and* t *or more of these shares can reconstruct the secret* S*. Also, equation (6.1) provides a correct procedure for checking the validity of each share.*

Proof. By the interpolation technique given in Section 5.5.1, there exists a unique PLF F such that $F(i) = S_i \in \mathcal{G}$ for $i \in \Phi$, where $\Phi \subset \{1, \ldots, n\}$ is a set such that $|\Phi| \geq t$.

Now, we show that the equation $\hat{e}(S_i, P) = \prod_{j=0}^{t-1} \alpha_j^{i^j}$ is correct: If each S_i is correct, we have

$$\begin{aligned} \prod_{j=0}^{t-1} \alpha_j^{i^j} &= \hat{e}(S, P) \prod_{j=1}^{t-1} \hat{e}(F_j, P)^{i^j} = \hat{e}(S, P) \prod_{j=1}^{t-1} \hat{e}(i^j F_j, P) \\ &= \hat{e}(S + iF_1 + \cdots + i^{t-1} F_{t-1}, P) = \hat{e}(S_i, P). \end{aligned}$$

□

The scheme `CVSSBM` is computationally secure in that the value $\alpha_0 = \hat{e}(S, P)$ is revealed during the execution of the protocol and hence the secrecy of S depends on the computational assumption that it is hard for an attacker to obtain S from $\hat{e}(S, P)$, which is actually the mGBI assumption presented in Definition 23. In the following, we formally prove this.

Lemma 10 *In* `CVSSBM`*, the attacker that learns fewer than* t *shares of the secret* S *obtains no information about* S *assuming that mGBI problem is computationally intractable.*

Proof. We show how an attacker for the mGBI problem $\mathsf{A}^{\mathsf{mGBI}}$ can be constructed using an attacker $\mathsf{A}^{\mathsf{CVSSBM}}$ for the `CVSSBM` scheme.

Without loss of generality, assume that the parties indexed $1, \ldots, t-1$ have been corrupted by $\mathsf{A}^{\mathsf{CVSSBM}}$. Also, assume that $\alpha_0 = \hat{e}(S, P)$ is provided as (public) input $\mathsf{A}^{\mathsf{mGBI}}$.

First, we pick $S_1, \ldots, S_{t-1}$ at random from $\mathcal{G}^*$ and compute $\alpha_1(= \hat{e}(S_1, P))$,..., $\alpha_{t-1}(= \hat{e}(S_{t-1}, P))$. We then compute $\alpha_t = \alpha_0^{\pi_{t0}} \prod_{j=1}^{t-1} \alpha_j^{\pi_{tj}}$, where π_{tj} denotes the Lagrange coefficient for an index set $\{0, 1, \ldots, t-1\}$. Subsequently, we give $S_1, \ldots, S_{t-1}$ (corrupted keys), $\alpha_1, \ldots, \alpha_{t-1}$ together with α_0 to $\mathsf{A}^{\mathsf{CVSSBM}}$. The simulated value α_t is correct and identically distributed to the one in the real execution of the `CVSSBM` scheme.

If $\mathsf{A}^{\mathsf{CVSSBP}}$ outputs its guess for the the secret value S, we return it as $\mathsf{A}^{\mathsf{mGBI}}$'s guess for the pre-image of α_0. □

6.4.4 Unconditionally Secure Verifiable Secret-Sharing Scheme Based on the Bilinear Map

Our second VSS scheme based on the Bilinear Map, which we call a "Unconditionally-secure VSS based on the Bilinear map (`UVSSBM`)", will be served as a basis for the new distributed key generation protocol based on the Bilinear map that will be described in Section 6.4.5. Our scheme is motivated by Pedersen's unconditionally (information-theoretically) secure VSS scheme [93], but due to the algebraic structure of the Bilinear map, it has some different features.

A New Commitment Scheme

We begin with describing a new commitment scheme, an important building block for the `UVSSBM` scheme.

Let $(\mathcal{G}, q, P, \hat{e})$ be the common parameters, as defined in Section 6.2.2. Suppose that random elements $G, H \in \mathcal{G}^*$, and the common parameters are given to a dealer. (We assume that no party knows $a, b \in \mathbb{Z}_q^*$ such that $G = aP$ and $H = bP$. These values can be chosen by a trusted third party or interested parties using a coin-flipping protocol.) The dealer then chooses $r \in \mathbb{Z}_q^*$ uniformly at random and computes

$$Comm(S, r) = \hat{e}(S, P)\hat{e}(G, H)^r.$$

The following lemma shows that the security of the above commitment scheme is relative to the CDH problem on the group $\mathcal{G}$.

Lemma 11 *If $Comm(S, r) = Comm(S', r')$ for $S, S' \in \mathcal{G}$, where $S \neq S'$, then we have $r \neq r'$ and find the DIffie-Hellman key of G and H.*

Proof. Let $Comm(S, r) = Comm(S', r')$. Since $G = aP$ and $H = bP$ for some $a, b \in \mathbb{Z}_q^*$, we have

$$\begin{aligned} \hat{e}(S, P)\hat{e}(G, H)^r &= \hat{e}(S', P)\hat{e}(G, H)^{r'} \\ \hat{e}(S, P)\hat{e}(aP, bP)^r &= \hat{e}(S', P)\hat{e}(aP, bP)^{r'} \\ \hat{e}(S - S', P) &= \hat{e}(aP, bP)^{r'-r} \\ \hat{e}(\frac{1}{r' - r}(S - S'), P) &= \hat{e}(abP, P). \end{aligned}$$

Hence, $\frac{1}{r'-r}(S - S')$ is the Diffie-Hellman key of G and H. □

Description of `UVSSBM`

Let $(\mathcal{G}, q, P, \hat{e})$ be a set of parameters, as defined in Section 6.2.2. Suppose that the threshold t and the number of parties n satisfy $1 \leq t \leq n < q$. To share a secret $S \in \mathcal{G}^*$ among n parties, a dealer performs the following:

1. Publish $\delta_0 = Comm(S, r)$, a commitment to S described previously, where r is chosen uniformly at random from $\mathbb{Z}_q^*$. (We assume that the dealer has used the random elements $G, H \in \mathcal{G}^*$ for input parameters for the commitment.)

2. Choose $F_1, \ldots, F_{t-1}$ uniformly at random from $\mathcal{G}^*$, construct a PLF $F(x) = S + xF_1 + \cdots + x^{t-1}F_{t-1}$ for $x \in \mathbb{N} \cup \{0\}$ and compute $S_i = F(i)$ for $i = 0, \ldots, n$. Set $S_0 = S$.

3. Choose $f_1, \ldots, f_{t-1}$ uniformly at random from $\mathbb{Z}_q^*$, construct a polynomial $f(x) = r + f_1 x + \cdots + f_{t-1}x^{t-1}$ for $x \in \mathbb{N} \cup \{0\}$ and compute $r_i = f(i)$ for $i = 0, \ldots, n$. Set $r_0 = r$.

4. Send (S_i, r_i) to party Γ_i for $i = 1, \ldots, n$ secretly. Broadcast $\delta_j = Comm(F_j, f_j)$ for $j = 1, \ldots, t - 1$.

5. Each party Γ_i then checks whether its share (S_i, r_i) is valid by computing

$$Comm(S_i, r_i) = \prod_{j=0}^{t-1} \delta_j^{ij}. \tag{6.2}$$

Analysis of `UVSSBM`

The `UVSSBM` scheme presented above satisfies correctness under the assumption that the CDH problem is intractable and unconditional secrecy. More precisely, we state and prove the following two lemmas.

Lemma 12 *In* `UVSSBM`*, shares held by all uncorrupted parties can be interpolated to a unique PLF of degree* $t-1$*, and* t *or more of these shares can reconstruct the secret* S*. Also, equation (6.2) provides a correct procedure for checking the validity of each share. In order to defeat the correctness of this procedure, a cheating attacker should solve the CDH problem.*

Proof. By the interpolation technique given in Section 5.5.1, there exists a unique PLF F such that $F(i) = S_i \in \mathcal{G}$ for $i \in \Phi$, where $\Phi \subset \{1, \ldots, n\}$ is a set such that $|\Phi| \geq t$. Also, by the interpolation technique in Shamir's secret-sharing scheme [107], there exists a unique polynomial f such that $f(i) = r_i \in \mathbb{Z}_q^*$ for $i \in \Phi$. Then, since $G = aP$ and $H = bP$ for some $a, b \in \mathbb{Z}_q^*$, we have

$$\begin{aligned} \hat{e}(F(i) + abf(i)P, P) &= \hat{e}(F(i), P)\hat{e}(abf(i)P, P) = \hat{e}(F(i), P)\hat{e}(aP, bP)^{f(i)} \\ &= Comm(S_i, r_i) = \hat{e}(S_i, P)\hat{e}(G, H)^{r_i} = \hat{e}(S_i + abr_iP, P) \end{aligned}$$

for $i \in \Phi$. Hence, $D(x) = F(x) + abf(x)P$ is the unique PLF that maps i to $S_i + abr_iP$. Indeed $D(x)$ is a PLF of degree $t-1$ since

$$\begin{aligned} D(x) &= F(x) + abf(x)P \\ &= S + xF_1 + \cdots + x^{t-1}F_{t-1} + uv(r + f_1x + \cdots + f_{t-1}x^{t-1})P \\ &= (S + abrP) + x(F_1 + abf_1P) + \cdots + x^{t-1}(F_{t-1} + abf_{t-1}P). \end{aligned}$$

Now, we show that the equation $Comm(S_i, r_i) = \prod_{j=0}^{t-1} \delta_j^{ij}$ is correct. Indeed, if each (S_i, r_i) is correct, we have $Comm(S_i, r_i) = \prod_{j=0}^{t-1} \delta_j^{ij}$ since

$$\prod_{j=0}^{t-1} \delta_j^{ij} = \hat{e}(S, P)\hat{e}(G, H)^r \prod_{j=1}^{t-1} (\hat{e}(F_j, P)\hat{e}(G, H)^{f_j})^{ij}$$

$$
\begin{aligned}
&= \hat{e}(S,P)\hat{e}(G,H)^r \prod_{j=1}^{t-1} (\hat{e}(i^j F_j, P)\hat{e}(G,H)^{f_j i^j}) \\
&= \hat{e}(S + iF_1 + \cdots + i^{t-1}F_{t-1}, P)\hat{e}(G,H)^{r+f_1 i + \cdots + f_{t-1} i^{t-1}} \\
&= \hat{e}(F(i),P)\hat{e}(G,H)^{f(i)} = \hat{e}(S_i,P)\hat{e}(G,H)^{r_i} = Comm(S_i, r_i).
\end{aligned}
$$

Moreover, as shown in Lemma 11, the chance that a cheating attacker can create a secret pair (S_i', r_i'), where $S_i' \neq S_i$ and $r_i' \neq r_i$, such that $Comm(S_i', r_i') = Comm(S_i, r_i)$ is negligible as the CDH problem on the group $\mathcal{G}$ is intractable. □

Lemma 13 *In* UVSSBM*, the attacker that learns fewer than t shares of the secret S obtains no information about S unconditionally.*

Proof. Without loss of generality, assume that the parties indexed $1, \ldots, t-1$ have been corrupted by an attacker $\mathsf{A}^{\mathsf{UVSSBM}}$ for UVSSBM . The attacker's view then consists of $\langle \delta_0, \delta_1, \ldots, \delta_{t-1}, (S_1, r_1), \ldots, (S_{t-1}, r_{t-1}) \rangle$. For any $S' \in \mathcal{G}$, there exists only one value $r' \in \mathbb{Z}_q^*$ satisfying $c_0 = Comm(S', r')$. Also, there exist only one PLF $F'(x) = S' + xF_1' + \cdots + x^{t-1}F_{t-1}'$ and only one polynomial $f'(x) = t' + f_1'x + \cdots + f_{t-1}'x^{t-1}$ such that $F'(0) = S'$ and $F'(i) = S_i$ for $i = 1, \ldots, t-1$, and $f'(0) = r'$ and $f'(i) = r_i$ for $i = 1, \ldots, t-1$.

But, by Lemma 12, there exists a unique PLF D of degree $t-1$ such that $\hat{e}(D(0), P) = Comm(S', r')$ and $\hat{e}(D(i), P) = Comm(S_i, r_i)$ for for $i = 1, \ldots, t-1$. Hence, we have $Comm(F_i', f_i') = \delta_i$ for for $i = 1, \ldots, t-1$, which implies that $\mathsf{A}^{\mathsf{UVSSBM}}$'s view does not contain any information about the secret. □

6.4.5 Distributed Key Generation Protocol Based on the Bilinear Map

Description of DKPBM

We are now ready to construct a distributed key generation protocol, whereby a number of parties *without* a dealer jointly generate a secret $K \in \mathcal{G}^*$ and its corresponding public value $\gamma = \hat{e}(K, P)$. We call this protocol a "Distributed Key generation Protocol Based on the Bilinear Map (DKPBM)". Notice that the aim of DKPBM is analogous to Gennaro et al.'s [60] distributed key generation protocol for discrete-logarithm based cryptographic schemes, whereby a predetermined set of

parties can generate a secret $k \in \mathbb{Z}_q^*$ and its corresponding public value $g^k \in \mathbb{Z}_p^*$ jointly, where g is a generator of $\mathbb{Z}_p^*$ and the primes p and q satisfy $q|p-1$.

To build DKPBM, we will need a distributed version of the UVSSBM scheme presented in Section 6.4.4, which we call a "Distributed Unconditionally-secure Verifiable secret-sharing scheme based on the Bilinear Map (DUVSSBM)". With this protocol, a secret $S \in \mathcal{G}^*$ can be generated jointly by participating parties without a dealer. Note that we adopt the description-style from [60] including some notations to describe the following protocols.

Suppose that a set of parameters $(\mathcal{G}, q, P, \hat{e})$ and a threshold t are given to all servers. Assume that the values $G, H \in \mathcal{G}^*$ used for the commitment scheme "*Comm*" are also given to n servers. Let $S \in \mathcal{G}^*$ be a secret to be shared among the n parties. The protocol DUVSSBM works as follows.

1. Each of the parties denoted by Γ_i performs the following.

   Choose $R_i, F_{i1}, F_{i2}, \ldots, F_{it-1}$ at random from $\mathcal{G}^*$; Choose $r_i, f_{i1}, f_{i2}, \ldots, f_{it-1}$ at random from $\mathbb{Z}_q^*$.

   Construct a PLF $F_i(x) = R_i + xF_{i1} + \cdots + x^{t-1}F_{it-1}$ and a polynomial $f_i(x) = r_{i0} + f_{i1}x + \cdots + f_{it-1}x^{t-1}$.

   Compute shares $S_{ij} = F_i(j) \in \mathcal{G}$ and $r_{ij} = f_i(j) \in \mathbb{Z}_q$ for $j = 1, \ldots, n$, and send (S_{ij}, r_{ij}) to Γ_j for each $j = 1, \ldots, n$.

   Broadcast $\delta_{i0} = Comm(R_i, r_{i0})$ and $\delta_{ik} = Comm(F_{ik}, f_{ik})$ for $k = 1, \ldots, t-1$.

2. Party Γ_j verifies the validity of the shares by checking

$$Comm(S_{ij}, r_{ij}) = \prod_{k=0}^{t-1} \delta_{ik}^{j^k} \tag{6.3}$$

   for $i = 1, \ldots, n$.

   If (6.3) fails for for some index i, Γ_j broadcasts a message $\mathsf{COMPLAINT}_i$ against Γ_i.

   If t or more parties have broadcast $\mathsf{COMPLAINT}_i$, party Γ_i is set as "*disqualified*". Otherwise, party Γ_i publishes all the values (S_{ij}, r_{ij}) that party Γ_j complained. That is, Γ_j publishes the values claimed to be failed (6.3) and the message $\mathsf{COMPLAINT}_i$ was created accordingly. Then, (6.3) is again performed for the published values by every party. If the values do not pass (6.3), party Γ_i is set to be *disqualified*.

3. Each party builds an index set of non-*disqualified* parties, denoted by QUAL.

4. Party Γ_i computes its share of the secret $S_i = \sum_{j \in \mathsf{QUAL}} S_{ji} \in \mathcal{G}$ and $r_i = \sum_{j \in \mathsf{QUAL}} r_{ji} \in \mathbb{Z}_q$. (Note that the distributed secret value S equals $\sum_{i \in \mathsf{QUAL}} R_i$ but no party can compute it explicitly.)

We now describe the DKPBM as follows.

1. Run DUVSSBM.

2. Party Γ_i, where $i \in \mathsf{QUAL}$, broadcasts $\beta_{i0} = \hat{e}(R_i, P)$ and $\beta_{ik} = \hat{e}(F_{ik}, P)$ for $k = 1, \ldots, t-1$.

3. Party Γ_j verifies the validity of the values broadcasted by the other parties that are qualified by checking

$$\hat{e}(S_{ij}, P) = \prod_{k=0}^{t-1} \beta_{ik}^{j^k} \tag{6.4}$$

   for each i such that $i \in \mathsf{QUAL}$.

   If (6.4) fails for for some index i, Γ_j broadcasts a $\widetilde{\mathsf{COMPLAINT}}_i$ against Γ_i, together with the pair of values (S_{ij}, r_{ij}) that satisfy (6.3) but do not satisfy (6.4).

4. For parties Γ_i that receive at least one pair of values (S_{ik}, r_{ik}) that satisfies (6.3) but does not satisfy (6.4), the other parties run the reconstruction phase of UVSSBM to compute R_i, $F_i(x)$ and β_{ik} for $k = 0, \ldots, t-1$. For all parities in QUAL, compute the verification information $\beta_k = \prod_{i \in \mathsf{QUAL}} \beta_{ik}$ for $k = 0, \ldots, t-1$. (Note that $\beta_0 = \hat{e}(S, P)$.)

Notice that the reason why the distributed version of "unconditionally secure" VSS scheme DUVSSBM (instead of a possibly more efficient distributed-VSS scheme based on the Bilinear map, which may be constructed in a similar way as Feldman's VSS scheme in [50]) is used to generate $S \in \mathcal{G}^*$ is that the attack on "Joint-Feldman VSS scheme" described in [60] is applied to DKPBM too.

Analysis of DKPBM

The correctness of DKPBM can be proven in a similar way as the scheme in [60]. We only show that β_0 in Step 4 of DKPBM does constitute $\hat{e}(S, P)$. This is indeed

the case since

$$\beta_0 = \prod_{j \in \mathsf{QUAL}} \beta_{j0} = \prod_{j \in \mathsf{QUAL}} \hat{e}(R_j, P) = \hat{e}(\sum_{j \in \mathsf{QUAL}} R_j, P) = \hat{e}(S, P).$$

In terms of security, DKPBM has interesting uniqueness in a sense that its security is relative to mGBI problem discussed in Section 6.2.2. We now state and prove the following lemma.

Lemma 14 *In* DKPBM*, the attacker that learns fewer than t shares of the secret S obtains no information about S assuming that mGBI problem is intractable.*

Proof. Suppose that the attacker $\mathsf{A}^{\mathsf{mGBI}}$ for the mGBI problem is given a random $\beta \in \mathcal{F}$. We show that $\mathsf{A}^{\mathsf{mGBI}}$ can construct a simulator providing the value $\beta \in \mathcal{F}$ as input. The aim of the simulator is to simulate an execution of DKPBM so that it looks indistinguishable from a real execution of DKPBM that outputs β as its public output β_0, where the DKPBM attacker $\mathsf{A}^{\mathsf{DKPBM}}$ possesses fewer than t shares of the secret S.

We now present details of the simulator. Assume that $\mathsf{A}^{\mathsf{DKPBM}}$ has corrupted parties $\Gamma_1, \ldots \Gamma_{t-1}$ without loss of generality. Let $\Phi_{cor} = \{1, \ldots, t-1\}$ be an index set of the corrupted parties. Let $\Phi_{uncor} = \{t, \ldots, n\}$ be an index set of the uncorrupted parties. Note here that $\Phi_{uncor} \subseteq \mathsf{QUAL}$. The simulator simulates each step of DKPBM as follows. (Remember that $\beta \in \mathcal{F}$ is provided as input to the simulator.)

> *Sim* 1. Run DUVSSBM for the uncorrupted parties $(\Gamma_t, \ldots, \Gamma_n)$. Let $\overline{F_i(x)} = \overline{R_i} + x\overline{F_{i1}} + \cdots + x^{t-1}\overline{F_{it-1}}$ and $\overline{f_i(x)} = \overline{r_i} + \overline{f_{i1}}x + \cdots + \overline{f_{it-1}}x^{t-1}$ for $i = 1, \ldots, n$ be the random PLF and polynomial respectively, held by each party. Let $(\overline{S_i}, \overline{r_i})$ denote the share held by each party at the end of of the protocol. (Note that all the PLFs, polynomials, $(\overline{S_i}, \overline{r_i})$ held by each party are known to the simulator.)
>
> *Sim* 2. Compute $\overline{\beta_{i0}} = \hat{e}(\overline{R_i}, P)$ and $\overline{\beta_{ik}} = \hat{e}(\overline{F_{ik}}, P)$ for $i \in \mathsf{QUAL} \setminus \{n\}$ and $k = 1, \ldots, t-1$. For n, set $\overline{\beta_{n0}} = \beta \prod_{i \in QUAL \setminus \{n\}} (\overline{\beta_{i0}})^{-1}$, compute $\overline{S_{nj}} = \overline{F_n(j)}$ for $j = 1, \ldots, t-1$ and $\overline{\beta_{nk}} = (\overline{\beta_{n0}})^{\pi_{k0}} \prod_{i=1}^{t-1} \hat{e}(\overline{S_{ni}}, P)^{\pi_{ki}}$ for $k = 1, \ldots, t-1$, where π_{ki} denotes a Lagrange interpolation coefficient for set $\{0, 1, \ldots, t-1\}$.
>
> Broadcast $\overline{\beta_{ik}}$ for $i \in \Phi_{uncor} \setminus \{n\}$, and $\overline{\beta_{nk}}$ for $k = 0, \ldots t-1$.

Sim 3, *Sim* 4 and *Sim* 5. Follow the executions of DKPBM for the uncorrupted parties.

□

6.5 Our ID-Based Threshold Signature Scheme

6.5.1 Description of Our ID-Based Threshold Signature Scheme

Combining the building blocks presented in the previous section, we now construct the ID-based threshold signature scheme based on the Bilinear map, which we call "IDTHSBM". IDTHSBM consists of the following algorithms. (For simplicity, we omit the details of sub-algorithms CVSSBM and DKPBM, and only describe the significant values resulted from them.)

- A key/common parameter generation algorithm $\mathsf{GC}(k)$: To generate the PKG's private and public key pair and all the necessary common parameters, the PKG performs the following.
  - Choose a group $\mathcal{G}$ of prime order q and its generator P. Specify the Bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{F}$.
  - Pick a master key s uniformly at random from $\mathbb{Z}_q^*$ and compute $P_{pub} = sP$.
  - Choose two hash functions $\mathsf{H}_1 : \{0,1\}^* \rightarrow \mathcal{G}$ and $\mathsf{H}_2 : \{0,1\}^* \times \mathcal{F} \rightarrow \mathbb{Z}_q^*$.
  - Keep s as secret and return $cp = (\mathcal{G}, q, P, P_{pub}, \hat{e}, \mathsf{H}_1, \mathsf{H}_2)$.
- A private key extraction algorithm $\mathsf{EX}(cp, \mathtt{ID})$: On receiving a private key extraction query ID from user, the PKG performs the following.
  - Compute $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ and $D_{\mathtt{ID}} = sQ_{\mathtt{ID}}$.
  - Return $D_{\mathtt{ID}}$.
- A private key distribution algorithm $\mathsf{DK}(cp, D_{\mathtt{ID}}, t, n)$: A trusted user (as discussed in Section 6.3.1, this user could be the PKG itself) who possesses a private key $D_{\mathtt{ID}}$ associated with an identity ID performs the following.

  - Run $\texttt{CVSSBM}$ taking $(\mathcal{G}, q, P, P_{pub}, \hat{e}, \mathsf{H}_1, t, n, D_{\texttt{ID}})$ as input to share $D_{\texttt{ID}}$ among n signature generation servers, denoted by $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$.
    * By $D^i_{\texttt{ID}}$ for $i = 1, \ldots, n$, denote each of the private key share of $D_{\texttt{ID}}$ held by Γ_i. By α_k for $k = 0, \ldots, t-1$, where t is a threshold, denote the public verification information output at the end of the execution of $\texttt{CVSSBM}$.

- A signature generation algorithm $\mathsf{S}(cp, D^i_{\texttt{ID}}, M)$ where $i = 1, \ldots, n$: Each signature generation server Γ_i performs the following to jointly generate a signature on a given message M.
  - Run $\texttt{DKPBM}$ taking $(\mathcal{G}, P, t, n)$ as input to jointly generate a secret value K and a public value $\gamma = \hat{e}(K, P)$.
    * Denote the resulting share of the sever Γ_i by K_i, where $i = 1, \ldots, n$. By $\beta_k = \prod_{i \in \mathsf{QUAL}} \beta_{ik}$ for $k = 0, \ldots, t-1$), denote the public verification information output at the end of the execution of $\texttt{DKPBM}$. Note that $\beta_0 = \hat{e}(K, P) = \gamma$.
  - Compute $v = \mathsf{H}_2(M, \gamma)$.
  - Broadcast $U_i = vD^i_{\texttt{ID}} + K_i$. (If Γ_i does not broadcast a value, we set U_i to *null*.)
  - For server Γ_i where $i \in \mathsf{QUAL}$, verify that

$$\hat{e}(U_i, P) = \Big(\prod_{k=0}^{t-1} \alpha_k^{i^k} \Big)^v \prod_{k=0}^{t-1} \beta_k^{i^k}. \tag{6.5}$$

  - Construct U by computing $U = \sum \pi_i U_i$ where π_i is the Lagrange coefficient for the set whose cardinality is at least t.
  - Return $\sigma = (U, v)$.

- A signature verification algorithm $\mathsf{V}(cp, \texttt{ID}, M, \sigma)$: Any user who wants to verify a signature $\sigma = (U, v)$ on a message M performs the following.
  - Compute $\gamma = \hat{e}(U, P)\hat{e}(Q_{\texttt{ID}}, -P_{pub})^v$, where $Q_{\texttt{ID}} = \mathsf{H}_1(\texttt{ID})$.
  - If $\mathsf{H}_1(M, \gamma) = v$ then return "*Accept*", otherwise return "*Reject*".

6.5.2 Remarks on Design

In fact, GC, EX, and V of IDTHSBM are the same as those of Hess' ID-based signature scheme (HessIDS) described in Section 6.4.1. Different are the private key distribution algorithm DK and signature generation algorithm S. We remark that during the execution of DK, the value $\hat{e}(D_{\mathtt{ID}}, P)$ is revealed as one of the public values of CVSSBM. However, this is not an additional value created aside from HessIDS since, from the values $P_{pub} = sP$ and $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$ which are publicly available in HessIDS, one can derive $\hat{e}(D_{\mathtt{ID}}, P)$ by computing $\hat{e}(Q_{\mathtt{ID}}, P_{pub}) = \hat{e}(Q_{\mathtt{ID}}, sP) = \hat{e}(sQ_{\mathtt{ID}}, P) = \hat{e}(D_{\mathtt{ID}}, P)$. Hence, the value $\hat{e}(D_{\mathtt{ID}}, P)$ resulted from CVSSBM does not affect the security of our ID-based threshold scheme. We also remark that although DK uses CVSSBM whose security is based on the mGBI problem (Lemma 10), the security of DK is not relative to the mGBI problem but the CDH problem since the values P_{pub} and $Q_{\mathtt{ID}}$ are given as additional inputs.

Note that although the validity of the shares of $D_{\mathtt{ID}}$ and K are checked during the executions of CVSSBM and DKPBM, whether the partial signatures on the message M do reconstruct the original signature is not ensured. To resolve this problem, we have adopted Cerecedo et al. [33]'s technique in which the publicly available values output CVSSBM and DKPBM are aggregated and the partial signatures are checked against them as presented in equation (6.5).

6.5.3 Variant for Non-Repudiation

One criticism on most ID-based signature schemes is that "non-repudiation", which is a very important property that signature schemes should possess, is not provided in the ID-based signature schemes due to the fact that the PKG always knows the user's private key and is capable of sign any messages at will.

As discussed in [66], the problem can be resolved by distributing the PKG's master key into a number of multiple PKGs, which is called the "distributed PKGs" method [27]. Our scheme IDTHSBM can incorporate the distributed PKGs technique as follows. First, the master key s is jointly generated by the multiple PKGs using the technique of [60]. Holding a share s_i of s, each of the multiple PKGs then responds to the trusted user who is supposed to run the private key distribution algorithm DK of IDTHSBM's private key extraction query with $D^i_{\mathtt{ID}} = s_i Q_{\mathtt{ID}}$ then the user collects these shares and recovers the private key $D_{\mathtt{ID}}$. Having recovered $D_{\mathtt{ID}}$, the user can run DK.

6.5.4 Security Analysis

We now analyze the security of IDTHSBM. First, recall that in order to show IDTHSBM is UF-IDTS-CMA secure, we need to show that the associated ID-based signature scheme HessIDS is UF-IDS-CMA secure and IDTHSBM is simulatable, as proven in Theorem 7. As mentioned before, the HessIDS scheme was actually proven to be UF-IDS-CMA secure assuming that the CDH problem is intractable in group $\mathcal{G}$ [66], hence, what we only have to prove is the following lemma.

Lemma 15 IDTHSBM *is simulatable.*

Proof. The simulator for the key distribution algorithm DK of IDTHSBM can be constructed in the same way as done in the proof of Lemma 10, which proves the security of the CVSSBM. However, we emphasize again that the security of DK of IDTHSBM is relative to the CDH problem (not mGBI) problem since the values P_{pub} and $Q_{\mathtt{ID}}$ are given as additional information to DK.

Now we present the simulator for the signature generation algorithm S of IDTHSBM. The simulator, given the PKG's public key P_{pub} and the parameter $cp = (\mathcal{G}, q, P, \hat{e}, \mathsf{H}_1, \mathsf{H}_2)$ generated by the key/common parameter generation algorithm GC of IDTHSBM, the identity ID and a signature (U, v) on message M, $D^1_{\mathtt{ID}}, \ldots, D^{t-1}_{\mathtt{ID}}$, which are $t-1$ shares of $D^i_{\mathtt{ID}}$, and the public outputs of DK, can simulate the view of the attacker on an execution of the signature generation algorithm S of IDTHSBM as follows.

Sim 0. Compute $\gamma^* = \hat{e}(U, P)\hat{e}(Q_{\mathtt{ID}}, -P_{pub})^v$, where $Q_{\mathtt{ID}} = \mathsf{H}_1(\mathtt{ID})$.

Sim 1. Do the following.

- Run DUVSSBP for the uncorrupted signature generation servers ($\Gamma_t, \ldots, \Gamma_n$). Let $\overline{F_i(x)} = \overline{R_i} + x\overline{F_{i1}} + \cdots + x^{t-1}\overline{F_{it-1}}$ and $\overline{f_i(x)} = \overline{r_i} + \overline{f_{i1}}x + \cdots + \overline{f_{it-1}}x^{t-1}$ for $i = 1, \ldots, n$ be the random PLF and polynomial respectively, held by each signature generation server. Let $(\overline{K_i}, \overline{r_i})$ denote the share held by each server at the end of of the protocol. (Note that all the PLFs, polynomials, $(\overline{K_i}, \overline{r_i})$ held by each server are known to the simulator.)
- Compute $\overline{\beta_{i0}} = \hat{e}(\overline{R_i}, P)$ and $\overline{\beta_{ik}} = \hat{e}(\overline{F_{ik}}, P)$ for $i \in \mathsf{QUAL} \setminus \{n\}$ and $k = 1, \ldots, t-1$. For n, set $\overline{\beta_{n0}} = \beta \prod_{i \in \mathsf{QUAL} \setminus \{n\}} (\overline{\beta_{i0}})^{-1}$, compute $\overline{K_{nj}} = \overline{F_n(j)}$ for $j = 1, \ldots t-1$ and $\overline{\beta_{nk}} = (\overline{\beta_{n0}})^{\pi_{k0}} \prod_{i=1}^{t-1} \hat{e}(\overline{K_{ni}}, P)^{\pi_{ki}}$

for $k = 1, \ldots, t-1$, where π_{ki} denotes a Lagrange coefficient for set $\{0, 1, \ldots, t-1\}$.
Broadcast $\overline{\beta_{ik}}$ for $i \in \Phi_{uncor} \setminus \{n\}$, and $\overline{\beta_{nk}}$ for $k = 0, \ldots, t-1$.

– Follow the instructions of the DKPBM protocol for the uncorrupted parties.

Sim 2. Compute $v = \mathsf{H}_2(M, \gamma^*)$.

Sim 3. Compute $U_i = vD_{\mathtt{ID}}^i + K_i$ for $i = 1, \ldots, t-1$. Let $F(u)$ be a PLF of degree $t-1$ such that $F(0) = U$ and $F(i) = U_i$ for $i = 1, \ldots, t-1$. Compute $U_i = F(i)$ for $i = t, \ldots, n$. Broadcast $U_t, \ldots, U_n$.

□

Combining Theorem 7, Lemma 15, the unforgeability of HessIDS from [66], and considering robustness, we obtain the following theorem.

Theorem 8 *The* IDTHSBM *is UF-IDTHS-CMA secure, relative to the CDH problem in the random oracle model.*

6.6 Brief Summary of the Results

In this chapter, we proposed the first ID-based threshold signature scheme based on the Bilinear map. We proved that our scheme is existentially unforgeable against chosen message attack, relative to the intractability of the Computational Diffie-Hellman problem. Like the ID-based threshold decryption scheme proposed in the previous chapter, our ID-based threshold signature scheme has the property that the private key issued by the PKG is shared among a number of signature generation servers rather than the master key of the PKG.

Chapter 7

Conclusion

From Chapter 3 through Chapter 6, we achieved the two main goals stated at the beginning of the thesis, (1) the analysis of the security of important existing cryptographic schemes through a new perspective and (2) the constructions of new cryptographic schemes that enhance the functionality of current public key cryptography.

In what follows, we summarize the contributions made in each chapter and discuss some problems that remained unsolved.

In Chapter 3, we showed that the slightly modified version of Zheng and Seberry's public key encryption scheme [129, 130], which had been proposed by Lim and Lee [77], is secure against chosen ciphertext attack in the random oracle model, relative to, in fact, the Gap Diffie-Hellman problem [90]. We also discussed the security analysis of Zheng and Seberry's scheme given by Soldera et al. [115] and concluded that their analysis is flawed.

Although our results in Chapter 3 will settle the issues related to the security of Zheng and Seberry's encryption scheme, raised in the series of the papers [77] and [115], it should be pointed out that our security analysis only holds in the random oracle model [20]. In fact, Zheng and Seberry's original intention was to design public key encryption schemes whose chosen ciphertext security can be proven without relying on non-standard assumptions (such as the random oracle model). In this respect, an interesting open problem is to weaken the assumptions used in the security analysis of Zheng and Seberry's scheme as much as possible. As envisioned by Zheng and Seberry [129, 130], the universal class of hash functions [31, 122] may be employed to replace some of the random oracles used in Zheng and Seberry's scheme (even though all the random oracles cannot be completely removed). In a broad respect, another hard but important open problem is to

design a public key encryption scheme provably secure against chosen ciphertext attack in the *standard* model (that is, the security model that does not depend on the random oracle assumption), which is *more efficient* than Cramer and Shoup's scheme [41].

In Chapter 4, we proved the confidentiality of Zheng's original signcryption scheme with respect to the strong security notion "FSO/FUO-IND-CCA" that we introduced. Although this notion bears some similarities to the well known "IND-CCA" notion defined for standard public key encryption schemes, it is stronger than the direct adaptation of IND-CCA to the setting of signcryption, since we allow an attacker to query both the signcryption oracle and the unsigncryption oracle in a flexible way. We also introduced a strong unforgeability notion called "FiSO-UF-CMA" and successfully proved the unforgeability of Zheng's original signcryption with respect to this notion.

We emphasize here that our security models for signcryption are applicable not only to Zheng's original scheme but also to other various signcryption schemes: As mentioned earlier in Chapter 4, Zheng's SDSS2-type signcryption scheme described in [124, 125] can be proven to be secure relative to the same computational assumptions for the SDSS1-type signcryption scheme using the same proof techniques presented in Chapter 4. Another immediate consequence of the results of this work is the provable confidentiality and unforgeability of the elliptic curve variants of Zheng's original signcryption scheme described in [126]. The only difference is that we need to rely on the intractability of elliptic curve versions of the GDH and DL problems in proving the security of those elliptic curve variants.

An interesting open problem in terms of the security analysis of Zheng's original signcryption scheme is to extend our security proof given in Chpater 4 to handle "adaptive security" in which the attacker chooses the target users dynamically. Another challenging open problem is to analyze Zheng's [127] signcryption scheme based on the high-residuosity problem. Since the common parameters of this scheme are generated differently from those of Steinfeld and Zheng's [116] signcryption scheme based on the Integer Factorization problem, new techniques other than those developed in [118] or [116] may be needed to analyze this scheme.

In Chapter 5, we discussed issues related to the realization of identity-based threshold decryption and proposed the first threshold identity-based decryption scheme provably secure against chosen ciphertext attack. An important feature of our scheme is that the *private key issued by the Private Key Generator (PKG) in identity-Based encryption is shared* among a number of decryption servers rather than the master key of the PKG, which, we claim, is more practical. We also showed how our identity-based threshold decryption scheme can result in a

mediated identity-based encryption scheme secure against inside attack, whereby an attacker who possesses a user part of private key conducts chosen ciphertext attack. The resulted scheme is actually the first mediated encryption scheme in the literature that possesses such a security property.

In terms of identity-based threshold decryption, an interesting open problem is to find more real-world applications where our identity-based threshold decryption scheme is particularly useful. However, we believe that a number of interesting open problems still remain in identity-based cryptography itself. Here is one of them: In identity-based cryptography, whenever users want to generate private keys associated with their identities, they should contact the PKG and ask for their private keys. Since this private key issuing process involves authentication of users' identifier information, it is quite a burden to users as well as to the PKG. Hence, a natural question one can ask is whether it is possible for users to acquire a single long-term private key from the PKG and use it to generate private keys for identities of their choice. Of course, in doing so, the system should remain secure against inside attackers, that is, cheating users, as well as outside attackers.

In Chapter 6, we formalized the concept of identity-based threshold signature and gave the first scheme realizing the concept, whose security is based on the Computational Diffie-Hellman problem. As building blocks for our identity-based threshold signature scheme, we constructed the various verifiable secrete-sharing schemes based on the Bilinear map. We expect that they as well as our identity-based threshold signature scheme will serve as sound primitives for identity-based cryptography in general.

An important fact that one should not overlook in relation to the research on identity-based cryptography is that Shamir [108] did successfully construct an identity-based signature scheme using the RSA primitive in his original proposal of identity-based cryptography. However, the recent identity-based signature variants e.g., [123, 76] including our scheme presented in Chapter 6, all use the Bilinear maps (pairings) as base primitives. In this respect, an interesting question is how to construct identity-based signature variants that achieve the same goals as those of the schemes in [123, 76] and/or Chapter 6 using the *conventional computational primitives* in which the Bilinear maps are *not* involved.

References

[1] M. Abdala, M. Bellare and P. Rogaway, *The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2001, Lecture Notes in Computer Science 2020, pages 143–158, Springer-Verlag, 2001.

[2] J. An, Y. Dodis and T. Rabin, *On the Security of Joint Signature and Encryption*, Advances in Cryptology - Proceedings of EUROCRYPT 2002, Lecture Notes in Computer Science 2332, pages 83–107, Springer-Verlag, 2002.

[3] R. Andersson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Inc., 2001.

[4] E. Artin, *Theory of Braids*, Annals of Math., No. 48, pages 101–126, 1947.

[5] J. Baek, B. Lee, and K. Kim, *Provably Secure Length-Saving Public-Key Encryption Scheme under the Computational Diffie-Hellman assumption* Electronics and Telecommunications Research Institute (ETRI) Journal, Vol. 22, No. 4, pages 25–31, 2000.

[6] J. Baek, R. Steinfeld, and Y. Zheng, *Formal Proofs for the Security of Signcryption*, Public Key Cryptography – Proceedings of PKC 2002, Lecture Notes in Computer Science 2274, pages 80-98, Springer-Verlag, 2002.

[7] J. Baek, R. Steinfeld, and Y. Zheng, *Formal Proofs for the Security of Signcryption*, Full-version, Submitted to Journal of Cryptology.

[8] J. Baek and Y. Zheng, *Zheng and Seberry's Public Key Encryption Scheme Revisited*, International Journal of Information Security (IJIS), Vol. 2, No. 1, pages 37–44, Springer-Verlag, 2003.

[9] J. Baek and Y. Zheng, *Simple and Efficient Threshold Cryptosystem from the Gap Diffie-Hellman Group*, IEEE Global Communications Conference

– Proceedings of IEEE GLOBECOM Conference, Communication Security Track, SC04-7, pages 1491–1495, IEEE, 2003.

[10] J. Baek and Y. Zheng, *Identity-Based Threshold Decryption*, Public Key Cryptography – Proceedings of PKC 2004, Lecture Notes in Computer Science, Springer-Verlag, 2004, to appear.

[11] J. Baek and Y. Zheng, *Identity-Based Threshold Signature from the Bilinear Pairings*, IEEE International Conference on Information Technology: Coding and Computing – Proceedings of ITCC 2004, Information Assurance and Security Track, IEEE Computer Society, 2004, to appear.

[12] F. Bao and R. Deng, *A Signcryption Scheme with Signature Directly Verifiable by Public Key*, Public Key Cryptography – Proceedings of PKC '98, Lecture Notes in Computer Science 1431, pages 55–59, Springer-Verlag, 1998.

[13] P. Barreto, H. Kim, B. Lynn, and M. Scott, *Efficient Algorithms for Pairing-Based Cryptosystems*, Advances in Cryptology - Proceedings of CRYPTO 2002, Lecture Notes in Computer Science 2442, pages 354–369, Springer-Verlag, 2002.

[14] M. Bellare, *Practice-Oriented Provable-Security*, Lectures on Data Security (Modern Cryptology in Theory and Practice), Lecture Notes in Computer Science 1561, pages 1–15, Springer-Verlag 1999.

[15] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, *A Concrete Security Treament of Symmetric Encryption*, IEEE Annual Symposium on the Foundations of Computer Science – Proceedings of FOCS '97, pages 394–403, IEEE Computer Society, 1997.

[16] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, Advances in Cryptology – Proceedings of CRYPTO '98, Lecture Notes in Computer Science 1462, pages 26–45 Springer-Verlag, 1998.

[17] M. Bellare and C. Namprepre, *Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm*, Advances in Cryptology – Proceedings of ASIACRYPT 2000, Lecture Notes in Computer Science 1976, pages 531–545, Springer-Verlag, 2000.

[18] M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Advances in Cryptology – Proceedings of EUROCRYPT '94, Lecture Notes in Computer Science 950, pages 92–111, Springer-Verlag, 1994.

[19] M. Bellare and P. Rogaway, *Exact Security of Digital Signatures - How to Sign with RSA and Rabin*, Advances in Cryptology – Proceedings of Eurocrypt '96, Lecture Notes in Computer Science 1070, pages 399–416, Springer-Verlag, 1996.

[20] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, ACM Conference on Computer and Communications Security – Proceedings of ACM CCCS '93, pages 62–73, ACM, 1993.

[21] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series 265, Cambridge University Press, 1999.

[22] D. Bleichenbacher, *Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1*, Advances in Cryptology – Proceedings of Crypto '98, Lecture Notes in Computer Science 1462, pages 1–12, Springer-Verlag, 1998.

[23] A. Biryukov, A. Shamir and D. Wagner, *Real-Time Cryptanalisis of A5/1 on a PC*, Proceedings of Fast Software Encryption 2000 (FSE 2000), Vol. 1978 of LNCS, Springer-Verlag 2000, pages 1–18.

[24] A. Boldyreva, *Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-group Signature Scheme*, Public Key Cryptography – Proceedings of PKC 2003, Lecture Notes in Computer Science 2567, pages 31–46, Springer-Verlag 2003.

[25] D. Boneh, *The Decision Diffie-Hellman Problem*, Algorithmic Number Theory Symposium – Proceedings of ANTS '98, Lecture Notes in Computer Science 1423, pages 48–63, Springer-Verlag, 1998.

[26] D. Boneh, X. Ding, G. Tsudik and C. Wong, *A Method for Fast Revocation of Public Key Certificates and Security Capabilities*, USENIX Security Symposium, Proceedings of USENIX Security Symposium, pages 297–308, USENIX, 2001.

[27] D. Boneh and M. Franklin, *Identity-Based Encryption from the Weil Pairing*, Advances in Cryptology – Proceedings of CRYPTO 2001, Lecture Notes in Computer Science 2139, pages 213–229, Springer-Verlag, 2001.

[28] D. Boneh, B. Lynn and H. Shacham, *Short Signatures from the Weil Pairing*, Advances in Cryptology – Proceedings of ASIACRYPT 2001, Lecture Notes in Computer Science 2248, pages 566–582, Springer-Verlag, 2001.

[29] R. Canetti and S. Goldwasser, *An Efficient Threshold Public Key Cryptosystem Secure Agaisnt Adaptive chosen-ciphertext Attack*, Advances in Cryptology – Proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pages 90–106, Springer-Verlag, 1999.

[30] R. Canetti, O. Goldreich and S. Halevi, *The Random Oracle Model, Revisited*, Annual ACM Symposium on Theory of Computing – Proceedings of STOC '98, pages 209–218, ACM, 1998.

[31] J. Carter and M. Wegman, *Universal Classes of Hash Functions*, Journal of Computer and System Sciences, Vol. 18, pages 143–154, Elsevier Inc., 1979.

[32] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann, *Factorization of a 512-bit RSA Modulus*, Advances in Cryptology – Proceedings of EUROCRYPT 2000, Lecture Notes in Computer Science 1807, pages 1–18, Springer-Verlag, 2000.

[33] M. Cerecedo, M. Matsumoto and H. Imai, *Efficient and Secure Multiparty Generation of Digital Signatures Based on Discrete Logarithms*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E76-A, pages 532–545, IEICE, 1993.

[34] J. Cha and J. Cheon, *An Identity-Based Signature from Gap Diffie-Hellman Groups*, Public Key Cryptography – Proceedings of PKC 2003, Lecture Notes in Computer Science 2567, pages 18–30, Springer-Verlag, 2003.

[35] D. Chaum, *Designated Confirmer Signatures*, Advances in Cryptology – Proceedings of CRYPTO '94, Lecture Notes in Computer Science 950, pages 86–91, Springer-Verlag 1995.

[36] D. Chaum and H. V. Antwerpen, *Undeniable Signatures*, Advances in Cryptology – Proceedings of CRYPTO '89, Lecture Notes in Computer Science 435, pages 212–216, Springer-Verlag 1990.

[37] D. Chaum and T. Perderson, *Wallet Databases with Observers*, Advances in Cryptology – Proceedings of CRYPTO '92, Lecture Notes in Computer Science 740, pages 89–105, Springer-Verlag, 1992.

[38] L. Chen, K. Harrison, D. Soldera, and N. P. Smart, *Applications of Multiple Trust Authorities in Pairing Based Cryptosysems*, International Conference on Infrastructure Security – Proceedings of InfraSec 2002, Lecture Notes in Computer Science 2437, pages 260–275, Springer-Verlag, 2002.

[39] C. Cocks, *An Identity Based Encryption Scheme Based on Quadratic Residues*, Cryptography and Coding - Institute of Mathematics and Its Applications International Conference on Cryptography and Coding – Proceedings of IMA 2001, Lecture Notes in Computer Science 2260, pages 360–363, Springer-Verlag, 2001.

[40] J. Coron, H. Handschuh, M. Joye, P. Paillier, D. Pointcheval and C. Tymen: *GEM: a Generic Chosen-Ciphertext Secure Encryption Method*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2002, Lecture Notes in Computer Science 2271, pages 263–276, Springer-Verlag, 2002.

[41] R. Cramer and V. Shoup: *A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack*, Advances in Cryptology – Proceedings of CRYPTO '98, Lecture Notes in Computer Science 1462, pages 13–25, Springer-Verlag, 1998.

[42] R. Cramer and V. Shoup: *Design and Analysis of Practical Public-Key Encryption Schemes against Adaptive Chosen Ciphertext Attack*, SIAM Journal on Computing, 2003, to appear.

[43] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung,*How to Share a Function Securely*, Annual ACM Symposium on Theory of Computing – Proceedings of STOC '94, pages 522–533, 1994.

[44] Y. Desmedt and Y. Frankel, *Threshold Cryptosystems*, Advances in Cryptology - Proceedings of CRYPTO '89, Lecture Notes in Computer Science 435, pages 307–315, Springer-Verlag, 1989.

[45] W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, Vol. 10, pages 74–84, IEEE, 1977.

[46] X. Ding and G. Tsudik, *Simple Identity-Based Cryptography with Mediated RSA*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2003, Lecture Notes in Computer Science 2612, pages 192–209, Springer-Verlag, 2003.

[47] Y. Dodis and M Yung, *Exposure-Resilience for Free: The Hierarchical ID-based Encryption Case.*, IEEE Security in Storage Workshop – Proceedings of SISW 2002, pages 45–52, IEEE Computer Society, 2002.

[48] D. Dolev, C. Dwork, and M. Naor, *Non-malleable Cryptography*, Annual ACM Symposium on Theory of Computing – Proceedings of STOC '91, pages 542–552, ACM, 1991.

[49] T. ElGamal: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol. 31, pages 469–472, IEEE, 1985.

[50] P. Feldman, *A Practical Scheme for Non-Interactive Verifiable Secret Sharing*, IEEE Annual Symposium on the Foundations of Computer Science – Proceedings of FOCS '87, pages 427–437, IEEE, 1987.

[51] A. Fiat and A. Shamir, *How to Prove Yourself: Practical Solutions of Identification and Signature Problems*, Proceedings of CRYPTO '86, Lecture Notes in Computer Science 263, pages 186–184, Springer-Verlag, 1987.

[52] P. Fouque and D. Pointcheval, *Threshold Cryptosystems Secure Chosen-Ciphertext Attacks*, Advances in Cryptology – Proceedings of ASIACRYPT 2001, Lecture Notes in Computer Science 2248, pages 351–368, Springer-Verlag, 2001.

[53] G. Frey, M. Müller, and H. Rück, *The Tate Paring and the Discrete Logarithm Appllied to Elliptic Curve Cryptosystems*, IEEE Transactions on Information Theory, Vol. 45, pages 1717–1718, IEEE Press, 1999.

[54] E. Fujisaki and T. Okamoto, *How to Enhance the Security of Public-Key Encryption at Minimum Cost*, Public Key Cryptography – Proceedings of PKC '99, Lecture Notes in Computer Science 1560, pages 53–68, Springer-Verlag, 1999.

[55] E. Fujisaki and T. Okamoto, *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, Advances in Cryptology – Proceedings of CRYPTO '99, Lecture Notes in Computer Science 1666, pages 537–554, Springer-Verlag, 1999.

[56] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, *RSA-OAEP is Secure under the RSA Assumption*, Journal of Cryptology, to appear.

[57] S. Galbraith, K. Harrison, and D. Soldera, *Implementing the Tate-pairing*, Algorithmic Number Theory Symposium – Proceedings of ANTS 2002, Lecture Notes in Computer Science 2369, pages 324–337, Springer-Verlag, 2002.

[58] C. Gamage, J. Leiwo, and Y. Zheng, *Encrypted Message Authentication by Firewalls*, Public Key Cryptography – Proceedings of PKC '99, Lecture Notes in Computer Science 1560, pages 69–81, Springer-Verlag, 1999.

[59] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, *Robust Threshold DSS Signatures*, Advances in Cryptology – Proceedings of EUROCRYPT '96, Lecture Notes in Computer Science 1070, pages 354–371, Springer-Verlag, 1996.

[60] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, *Secure Distributed Key Generation for Discrete-Log Based Cryptosystem*, Advances in Cryptology – Proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pages 295–310, Springer-Verlag, 1999.

[61] C. Gentry and A. Silverberg, *Hierarchical ID-Based Cryptography*, Advances in Cryptology – Proceedings of ASIACRYPT 2002, Lecture Notes in Computer Science 2501, pages 548–566, Springer-Verlag, 2002.

[62] S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, pages 270–299, Elsevier Inc., 1984.

[63] S. Goldwasser, S. Micali and R. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*, SIAM Journal on Computing, Vol. 17, No. 2, pages 281–308, Society for Industrial and Applied Mathematics, 1988.

[64] F. Grieu, *A Chosen Messages Attack on the ISO/IEC 9796-1 Signature Scheme*, Advances in Cryptology – Proceedings of EUROCRYPT 2000, Lecture Notes in Computer Science 1807, pages 70–80, Springer-Verlag, 2000.

[65] G. Hanaoka and Y. Zheng, *LITESET: A Light-Weight Secure Electronic Transaction Protocol*, Australasian Conference on Information Security and Privacy – Proceedings of ACISP '98, Lecture Notes in Computer Science 1438, pages 215–226, Springer-Verlag, 1998.

[66] F. Hess, *Efficient Identity Based Signature Schemes Based on Pairings*, Selected Areas in Cryptography – Proceedings of SAC 2002, Lecture Notes in Computer Science 2595, pages 310–324, Springer-Verlag, 2002.

[67] IEEE P1363, Standard Specifications for Public Key Cryptography, 2000.

[68] M. Jakobsson and D. Pointcheval, *Mutual Authentication for Low-Power Mobile Devices*, Financial Cryptography – Proceedings of FC 2001, Lecture Notes in Computer Science 2339, pages 169–186, Springer-Verlag, 2001.

[69] A. Joux, *The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems*, Algorithmic Number Theory Symposium – Proceedings of ANTS 2002, Lecture Notes in Computer Science 2369, pages 20–32, Springer-Verlag, 2002.

[70] C. Jutla, *Encryption Modes with Almost Free Message Integrity*, Advances in Cryptology - Proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science 2045, pages 529–544, Springer-Verlag, 2001.

[71] A. Khalili, J. Katz, and W. Arbaugh, *Toward Secure Key Distribution in Truly Ad-Hoc Networks*, IEEE Symposium on Applications and the Internet Workshops – Proceedings of SAINT 2003, pages 342–346, IEEE Computer Society, 2003.

[72] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, and C. Park, *New Public-Key Cryptosystem Using Braid Groups*, Advances in Cryptology – Proceedings of CRYPTO 2000, Lecture Notes in Computer Science 1880, pages 166–183, Springer-Verlag, 2000.

[73] N. Koblitz, *Elliptic Curve Cryptosystems*, Mathematics of Computation, Vol. 48, No. 177, pages 203–209, American Mathematical Society, 1987.

[74] H. Krawczyk, *The Order Of Encryption And Authentication For Protecting Communications (Or: How Secure Is SSL?)*, Advances in Cryptology - Proceedings of CRYPTO 2001, Lecture Notes in Computer Science 2139, pages 310–331, Springer-Verlag, 2001.

[75] B. Libert and J. Quisquater, *Efficient Revocation and Threshold Pairing Based Cryptosystems*, ACM Symposium on Principles of Distributed Computing – Proceedings of PODC 2003, pages 163–171, ACM, 2003.

[76] B. Libert and J. Quisquater, *Identity Based Undeniable Signatures*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2004, Lecture Notes in Computer Science ????, Springer-Verlag, 2004, to appear.

[77] C. Lim and P. Lee, *Another Method for Attaining Security Against Adaptively chosen ciphertext Attack*, Advances in Cryptology - Proceedings of CRYPTO '93, Lecture Notes in Computer Science 773, pages 410–434, Springer-Verlag, 1993.

[78] W. Mao, *Modern Cryptography: Theory & Practice*, Prentice Hall, 2004.

[79] K. Matsuura and H. Imai, *Toward Research-Promotion Infrastructure for Multi-Modal Imaging*, Recent Advances in Biomagnetism, pages 911–914, Tohoku University Press, 1999.

[80] U. Maurer, *Towards Proving the Equivalence of Breaking the Diffie-Hellman Protocol and Computing Discrete Logarithms*, Advances in Cryptology – Proceedings of CRYPTO '94, Lecture Notes in Computer Science 839, pages 271–281, Springer-Verlag, 1994.

[81] A. J. Menezes, T. Okamoto, and S. A. Vanstone: *Reducing Elliptic Curve Logarithms to a Finite Field*, IEEE Transactions on Information Theory, Vol. 31, pages 1639–1646, IEEE, 1993.

[82] V. Miller, *Use of Elliptic Curves in Cryptography*, Advances in Cryptology – Proceedings of CRYPTO '85, Lecture Notes in Computer Science 218, pages 417–429, Springer-Verlag, 1986.

[83] R.J. McEliece, *A Public-Key Cryptosystem Based on Algebraic Coding Theory*, Deep Space Network Progress Report 42-44, pages 114–116, Jet Propulsion Lab., California Institute of Thchnology, 1978.

[84] National Breau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS Publication 46, NBS, 1977.

[85] M. Naor and M. Yung, *Public-Key Cryptosystems Provably Secure against chosen ciphertext Attacks*, Annual ACM Symposium on Theory of Computing – Proceedings of STOC '90, pages 427– 437, ACM, 1990.

[86] National Institute of Standards and Technology, *Digital Signature Standards*, U.S. Department of Commerce, NIST FIPS PUB 186, May 1994.

[87] National Institute of Standards and Technology, *Secure Hash Standar*, U.S. Department of Commerce, NIST FIPS PUB 180-1, April 1995.

[88] K. Ohta and T. Okamoto, *On Concrete Security Treatment of Signatures Derived from Identification*, Advances in Cryptology – Proceedings of CRYPTO '98, Lecture Notes in Computer Science 1462, pages 354–369, Springer-Verlag, 1998.

[89] T. Okamoto and S. Uchiyama, *A New Public-Key Cryptosystemas as Secure as Factoring*, Advances in Cryptology – Proceedings of EUROCRYPT '98, Lecture Notes in Computer Science 1403, pages 308–318, Springer-Verlag, 1998.

[90] T. Okamoto and D. Pointcheval, *The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes*, Public Key Cryptography – Proceedings of PKC 2001, Lecture Notes in Computer Science 1992, pages 104–118, Springer-Verlag, 2001.

[91] T. Okamoto and D. Pointcheval, *REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2001, Lecture Notes in Computer Science 2020, pages 159–174, Springer-Verlag, 2001.

[92] P. Paillier, *Public Key Cryptosystems Based on Composite Degree Redidu-osity Classes*, Advances in Cryptology - Proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pages 223–238, Springer-Verlag, 1999.

[93] T. P. Pedersen, *Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing*, Advances in Cryptology - Proceedings of CRYPTO '91, Lecture Notes in Computer Science 576, pages 129–140, Springer-Verlag, 1992.

[94] M. Pistoia, D. Reller, D. Gupta, M. Nagnur, and A. Ramani, *Java 2 Network Security*, IBM Redbook Series, Prentice Hall, 1999.

[95] D. Pointcheval, *Chosen-Ciphertext Security for Any One-Way Cryptosystem*, Public Key Cryptography – Proceedings of PKC 2000, Lecture Notes in Computer Science 1751, pages 129–146, Springer-Verlag, 2000.

[96] D. Pointcheval, *New Public Key Cryptosystems Based on the Dependent-RSA Problems*, Advances in Cryptology – Proceedings of EUROCRYPT '99, Lecture Notes in Computer Science 1592, pages 239–254, Springer-Verlag, 1999.

[97] D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures*, Journal of Cryptology, Vol. 13, No. 3, pages 361-396, Springer-Verlag, 2000.

[98] B. Preneel et al., *Security Evaluation of NESSIE First Phase*, New European Schemes for Signature, Integrity and Encryption (NESSIE) project report (IST-1999-12324), Full version available at `http://www.cryptonessie.org/`.

[99] M.O. Rabin, *Digitalized Signatures and Public-Key Functions as Intractable as Factorization*, MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[100] C. Rackoff and D. Simon, *Noninteractive Zero-Knowledge Proof of Knowledge and chosen ciphertext Attack*, Advances in Cryptology - Proceedings of CRYPTO '91, Lecture Notes in Computer Science 576, pages 434–444, Springer-Verlag, 1992.

[101] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, *OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption*, ACM Conference on Computer and Communications Security – Proceedings of ACM CCCS 2001, pages 196–205, ACM, 2001.

[102] Ronald L. Rivest. *The MD5 Message-Digest Algorithm*, Internet Repost, RFC 1321, April 1992.

[103] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM 21 (2), pages 120–126, 1978.

[104] C. P. Schnorr, *Efficient Identification and Signatures for Smart Cards*, Advances in Cryptology - Proceedings of CRYPTO '89, Lecture Notes in Computer Science 435, pages 235–251, Springer-Verlag, 1990.

[105] C. P. Schnorr and M. Jakobsson, *Security of Signed ElGamal Encryption* , Advances in Cryptology - Proceedings of ASIACRYPT 2000, Lecture Notes in Computer Science 1976, pages 73–89, Springer-Verlag, 2000.

[106] M. Seo and K. Kim, *Electronic Funds Transfer Protocol Using Domain-verifiable Signcryption Scheme*, International Conference on Information Security and Cryptology – Proceedings of ICISC '99, Lecture Notes in Computer Science 1787, pages 269–277, Springer-Verlag, 1999.

[107] A. Shamir, *How to Share a Secret*, Communications of the ACM, Vol. 22, pages 612–613, 1979.

[108] A. Shamir, *Identity-based Cryptosystems and Signature Schemes*, Advances in Cryptology - Proceedings of CRYPTO '84, Lecture Notes in Computer Science 196, pages 47–53, Springer-Verlag, 1984.

[109] V. Shoup, *A Proposal for an ISO Standard for Public Key Encryption (Version 2.1)*, ISO/IEC JTC 1/SC 27, 2001.

[110] V. Shoup, *OAEP Reconsidered*, Advances in Cryptology – Proceedings of CRYPTO 2001, Lecture Notes in Computer Science 2139, pages 239–259, Springer-Verlag, 2001.

[111] V. Shoup and R. Gennaro, *Securing Threshold Cryptosystems against chosen ciphertext Attack*, Advances in Cryptology – Proceedings of EUROCRYPT '98, Lecture Notes in Computer Science 1403, pages 1–16, Springer-Verlag, 1998.

[112] V. Shoup and R. Gennaro, *Securing Threshold Cryptosystems against chosen ciphertext Attack*, Journal of Cryptology, Vol. 15, No. 2, pages 75–96, Springer-Verlag, 2002.

[113] N. P. Smart, *Access Control Using Pairing Based Cryptography*, Topics in Cryptology: Cryptographer's Track in RSA Conference – Proceedings of CT-RSA 2003, Lecture Notes in Computer Science 2612, pages 111–121, Springer-Verlag, 2003.

[114] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing Company, 1997.

[115] D. Soldera, J. Seberry, and C. Qu, *The analysis of Zheng-Seberry scheme*, Australasian Conference on Information Security and Privacy – Proceedings of ACISP 2002, Lecture Notes in Computer Science 2384, pages 159–168, Springer-Verlag, 2002.

[116] R. Steinfeld and Y. Zheng, *A Signcryption Scheme Based on Integer Factorization*, Information Security Workshop – Proceedings of ISW 2000, Lecture Notes in Computer Science 1975, pages 308–322, Springer-Verlag, 2000.

[117] R. Steinfeld, J. Baek and Y. Zheng, *On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes*, Australasian Conference on Information Security and Privacy – Proceedings of ACISP 2002, Lecture Notes in Computer Science 2384, pages 241–256, Springer-Verlag, 2002.

[118] R. Steinfeld, *Analysis and Design of Public-Key Cryptographic Schemes*. PhD Thesis, Monash University, January 2003.

[119] D. Stinson, *Cryptography: Theory and Practice*, 2nd Ed., Chapman & Hall/CRC, 2002.

[120] D. Stinson and R. Strobl, *Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates*, Australasian Conference on Information Security and Privacy – Proceedings of ACISP 2001, Lecture Notes in Computer Science 2119, pages 417–434, Springer-Verlag, 2001.

[121] Y. Tsiounis and M. Yung, *On the Security of ElGamal-Based Encryption*, Public Key Cryptography – Proceedings of PKC '98, Vol. Lecture Notes in Computer Science 1431, pages 117–134, Springer-Verlag, 1998.

[122] M. Wegman and J. Carter, *New Hash Functions and Their Use in Authentication and Set Equality*, Journal of Computer and System Sciences, Vol. 22, pages 265–279, Elsevier Inc., 1981.

[123] F. Zhang and K. Kim, *ID-based Blind Signature and Ring Signature from Pairings*, Advances in Cryptology – Proceddings of ASIACRYPT 2002, Lecture Notes in Computer Science 2501, pages 533–547, Springer-Verlag, 2002.

[124] Y. Zheng, *Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost (Signature) + Cost (Encryption)*, Advances in Cryptology - Proceedings CRYPTO '97, Lecture Notes in Computer Science 1294, pages 165–179, Springer-Verlag, 1997.

[125] Y. Zheng, *Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost (Signature) + Cost (Encryption)*, full version, available at `http://www.sis.uncc.edu/~yzheng/papers/`.

[126] Y. Zheng and H. Imai, *Efficient Signcryption Schemes On Elliptic Curves*, International Information Security Conference – Proceedings of IFIP/SEC '98, pages 75–84, Chapman and Hall, 1998.

[127] Y. Zheng, *Identification, Signature and Signcryption Using High Order Residues Modulo an RSA Composite*, Public Key Cryptography – Proceedings of PKC 2001, Lecture Notes in Computer Science 1992, pages 48–63, Springer-Verlag, 2001.

[128] Y. Zheng, *Improved Public Key Cryptosystems Secure against chosen ciphertext Attacks*, Technical Note, The Centre for Computer Security Research, University of Wollongong, 1994.

[129] Y. Zheng and J. Seberry, *Immunizing Public Key Cryptosystems against chosen ciphertext Attacks*, Special Issue on Secure Communications, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, pages 715–724, 1993.

[130] Y. Zheng and J. Seberry, *Practical Approaches to Attaining Security against Adaptively chosen ciphertext Attacks*, Advances in Cryptology – Proceedings of CRYPTO '92, Lecture Notes in Computer Science 742, pages 292–304, Springer-Verlag, 1993.