GENERIC CONSTRUCTIONS OF DIFFERENT CRYPTOGRAPHIC PRIMITIVES OVER VARIOUS PUBLIC KEY PARADIGMS

Thesis submitted to Indian Statistical Institute

by

Sumit Kumar Pandey

2014

GENERIC CONSTRUCTIONS OF DIFFERENT CRYPTOGRAPHIC PRIMITIVES OVER VARIOUS PUBLIC KEY PARADIGMS

Thesis submitted to Indian Statistical Institute in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy

by

Sumit Kumar Pandey
CR RAO AIMSCS
UOH Campus, Gachibowli
Hyderabad - 500046, INDIA

under the supervision of

Prof. Rana Barua
Stat-Math Unit
Indian Statistical Institute
203, B.T. Road, Kolkata - 700108, INDIA

Acknowledgements

With great pleasure and sense of obligation I express my heartfelt gratitude to my guide and supervisor Prof. Rana Barua. I am highly indebted to him for their invaluable guidance and ever ready support. Their persisting encouragement, perpetual motivation, everlasting patience and excellent expertise in the field of knowledge, have benefited to an extent, which is beyond expression.

Several people have inspired and motivated me in these long years. I would like to express my gratitude to Dr. Kishan Chand Gupta of ASU and Dr. Mahavir Prasad Jhanwar for their constant support and valuable suggestions given to me throughout my research work. My sincere thanks to Dr. Mahavir Prasad Jhanwar who spared his valuable time to teach me some basic lessons on cryptography and motivated me to work in my research field.

I express my gratitude to Dr. Suchismita Roy and Soumyottam Chatterjee who never left me alone when I was in distress for a long period. If they were not in my life, I would have quit my research career and done something else. They always stood by me during both good and bad times.

I completed my research work as a research scholar at Applied Statistical Unit (ASU), Indian Statistical Institute (ISI), Kolkata. I would like to thank Indian Statistical Institute for financial and institutional support. I also thank to all my friends and all members of Applied Statistics Unit, Indian Statistical Institute for their friendly interactions and to all the staff of ASU for providing all kinds of facilities whenever required.

I must say thanks to all research scholars, faculties and staffs of CR RAO Advanced Institute of Mathematics, Statistics and Computer Science (AIMSCS), Hyderabad for their constant support and encouragement. They always helped me not only like my friends or colleagues but also like my family members to reduce my stress whenever I felt alone and distressed.

Finally, I express my heartiest thanks to my parents and my family members for their love and encouragement.

Contents

Chapter 1

Introduction

In this thesis, we study the generic construction of some cryptographic primitives over various public key paradigms like traditional Public Key Cryptosystems and Identity Based Cryptosystems. It can be broadly divided into two categories-

1. Generic construction of some highly secure cryptographic primitives from less secure cryptographic primitives, and

2. Generic construction of some complex cryptographic primitives from basic cryptographic primitives.

Mathematical tools provide a way to achieve cryptographic functionality like confidentiality, authentication, data-integrity, non-repudiation etc., but in the case of complex cryptographic functionality like achieving confidentiality and authentication at the same time or confidentiality, authentication and non-repudiation at the same time etc., proper combination of basic cryptographic tools is desired. Achieving complex cryptographic primitives using mathematical tools directly leads to a tedious job whereas breaking the combination of functionalities into smaller and basic functionalities and then using the basic primitives and then joining them in a proper manner for achieving the desired functionality is somewhat more methodical approach which, in result, helps in analysing the security in better and easier manner. Being a more methodical way, it has not only the theoretical importance but also has a vast practical utility. It can be analogously compared as constructing a structure using bricks, cement and steel rods.

Public Key cryptosystem took birth from the seminal paper by Diffie and Hellman [36] in the year 1976. It is also known as Asymmetric Key cryptosystem due to the nature of keys. In this cryptosystem, both parties which are communicating have different keys in contrast to Symmetric Key cryptosystem in which both parties have the same key. Although one key exchange protocol was proposed based on one hard assumption that is known as computational Diffie-Hellman assumption derived from the names of proposer, no public key scheme was proposed. Crypto community had to wait till 1978 when Rivest, Shamir and Adleman (RSA) [81] proposed the first public key encryption (PKE) and signature (PKS) scheme which is popularly known as RSA encryption and RSA signature scheme respectively which were based upon another hard assumption which is known as RSA assumption. In the same

year, McEliece [70] proposed another PKE using Goppa code which is known as McElice encryption scheme based on McEliece assumption. But the simplicity of RSA gave it much more popularity than McElice. Again in the year 1985, ElGamal [39] proposed another PKE based on computational Diffie-Hellman assumption which is known as ElGamal encryption scheme. Since then, many encryption and signature schemes have been proposed. In commercial purpose protocols like secure socket layer (SSL) [42], PKE is used for exchanging the session keys. Amongst all public key encryption schemes, RSA is the most popular and widely used.

The issue of online key exchange, an essential step in the symmetric key cryptosystem, was solved by the advent of public key cryptosystem, but it required a proper key management [4]. In the year 1984, to simplify key management, Shamir proposed Identity based cryptosystem [85]. In this cryptosystem, the unique identity (such as email-id, social security number etc.) of user is used as the public key whereas the secret key is generated by a trusted third party called Private Key Generator (PKG). Along with the proposal of Identity Based cryptosystem, Shamir proposed one Identity Based Signature (IBS) scheme also [85]. But, it was a long journey for Identity Based Encryption (IBE) scheme when, in the year 2001, Sakai, Ohghishi & Kasara [83] and Boneh & Franklin [15] proposed the first IBE scheme independently based on bilinear maps. Same year, Clifford Cocks [29] solved the problem of constructing an IBE without bilinear maps. His scheme was based on the quadratic residuosity assumption. Since then, many IBE [91, 22, 5, 45, 71, 16, 56, 57] and IBS [78, 27, 7, 63] schemes have been proposed.

1.1 Need for Security Notions

The idea of security is the central point of cryptography. The design of a cryptographic scheme is followed by either security arguments (heuristic) or security proofs. Thus the notion of security must be clearly defined. It is defined with two important parameters - (a) goal of adversary and (b) type of attack. *Goal of adversary* is to break the functionality of cryptographic scheme. Depending upon the purpose and sensitive demand of the application, the notion of security changes [9, 10, 20]. For example, for general user, leakage of the message partially may not cause a severe damage but in defence, it may be a serious issue. Hence, a cryptographic scheme (in this example, encryption scheme) which guarantees the hiding of complete message (not partial) may be useful for general user but not for defence. Ultimately, varied scenarios lead to different goals of adversary.

The second important factor for defining security notions is *type of attack* [9, 10, 20]. It refers to the amount of information an adversary may have. The simplest scenario is when adversary does not have any extra information other than the ciphertext. But, it does not capture the full picture. For example, an adversary may trap many ciphertexts from the communication channel and hence have extra information. Not only that, sometimes, he/she can have access to the machine like encryption machine, decryption machine, signature machine etc. and can use this (these) for a period of time for his/her own purpose. Henceforth, a proper analysis of type of attacks is required while designing any cryptographic scheme for

given scenario. In brief, where *goal of adversary* is to *break the functionality* of cryptographic scheme, *type of attack* refers to the amount of information adversary may gather. Thus, the need of security notion becomes obvious as it gives the correct view of the security achieved by the proposed cryptographic scheme.

The security of a protocol depends not only on the cryptographic primitives but also how they are combined. Improper combination of even highly secure cryptographic primitives may result in a less secure protocols; whereas even less cryptographic primitives may give a highly secure protocols [20]. In theory, it is always desirable to use these primitives which achieve the highest level of security, but in practice, apart from security there are two other important aspects - communication and computational overhead - which run parallel to the security. Those protocols or cryptographic schemes are practically of no use whose computational or communication overhead is exponential. It is an observational fact (not mathematical) that as the level of security increases, computational and communication overhead increase [81, 39, 73, 31, 12, 30]. Hence, to seek for the exact security requirement is a better choice than choosing always the most secure cryptographic primitives.

The security requirement of a protocol or cryptographic primitive is modelled mathematically and then it is analysed to check the achieved security level [20]. The mathematical model should simulate the real threat model as close as possible. Security model of any cryptographic primitive is modelled as a game between an adversary and a challenger [8]. Depending upon the the real threat, in the security model (notion), the adversary is provided some oracle which responds in the constant time for each query given by the receiver [8, 10]. Some models (notions) put restriction upon the type of queries [20, 53, 79] also. In brief, (a) behaviour of oracle and (b) restriction upon queries give the structure for *type of attack*.

By behaviour of oracle, we mean whether it is computational or decisional. By computational we mean, the response of oracle is neither *true* nor *false* (1/0), but it simulates any computational machine like *Decryption* or *Signature* or *Signcryption* or *Designcryption* machine etc. We raise the question and ask what if the behaviour of oracle is decisional - response is either *true* or *false*. Such behaviour of oracle is not new in the literature like *plaintext checking* [74] oracle which is decisional in nature. Given a message-ciphertext pair to the *plaintext checking* oracle, it returns *true* (1) if that message is the decryption of the given ciphertext else it returns *false* (0). In a similar fashion, we asked a natural question - "What happens when oracle decides whether a given ciphertext is a valid ciphertext or an invalid ciphertext"? We name this oracle, *Chosen Ciphertext Verification* Oracle. Exploring the answer to the above mentioned question, we encounter one new security notion, IND-CCVA, and we show that it lies between IND-CPA and IND-CCA2. Although it appears that this oracle is a completely theoretical oracle, it is quite not true. Bleichenbacher [14] has shown an attack on PKCS #1 using *Chosen Ciphertext Verification* Oracle supplying full relevance in favour of such notion. Chapter 3 deals with this new security notion where formal definition of IND-CCVA and its relation with existing security notions like IND-CPA, IND-CCA1 and IND-CCA2 have been discussed.

1.2 Combination of Different Functionalities

Cryptography is not new to the world. From the time of Julius Ceaser, it has been used as an art of hiding the secret [87]. Confidentiality was the main functionality, but as time grew, its goal has become wider. Other functionalities like authentication, data-integrity and non-repudiation have been added into the list [87]. Now, when the application is not limited to a specific role, these functionalities alone are not sufficient. However, the combination of functionalities in many cases give the desired solution [12, 8, 94, 21, 18, 43]. For example, the security of mail does not rely upon the confidentiality of message only but also upon the data-integrity and authentication of the sender. Although simple patching of different schemes corresponding to different functionalities may be one of the solutions, yet the desired security may not be achieved. In case of achieving confidentiality and authentication, three different and obvious paradigms are (a) sign then encrypt (b) encrypt then sign and (c) encrypt and sign. Although all paradigms yield confidentiality and authentication both, latter two paradigms do not provide the desired security [3]. Keeping view of the needs of applications, the proper study and analysis of different combinations of various cryptographic primitives become essential to provide both security and efficiency.

Use of simple cryptographic primitives is one of the two approaches to build complex cryptographic primitives. Another approach is to use a set of assumed hard mathematical problems and then to use this set to construct desired schemes. Efficiency is the main advantage of the later approach, but as the complexity of scheme grows, building such scheme becomes a tedious task. Where the construction from the former approach gives many instantiations of the scheme, later approach gives only one construction at a time. Besides it, the former approach has two more advantages - (a) easier to construct (b) easier to do security analysis. The former is in general less efficient than the later approach. However, the former approach gives a much more clear understanding about the behaviour of simple cryptographic primitives and its behaviour with the built complex primitives.

Following the first approach, in this thesis, we give (a) two different generic constructions of Identity Based Signcryption schemes using Identity based Encryption and Identity based Signature schemes (b) two different generic constructions of Signcryption schemes from Encryption and Signature schemes and (c) a new variant of Ring Signature called Ring Signature with Designated Verifier for Signer's Identity and its generic construction using Ring Signature and Identity Based Signcryption schemes. There are paradigms to construct one signcryption scheme like Commit-then-Encrypt-and-Sign paradigm [3] which can be lifted to construct Identity Based Signcryption schemes. There are other generic constructions also [26, 69] for constructing signcryption schemes using Tag-Based Key Encapsulation Mechanisms or Data Encapsulation Mechanisms [69] which give efficient constructions. Our proposed constructions of signcryption schemes or identity based signcryption schemes may not be efficient than many existing schemes. However, these are very much efficient constructions using encryption and signature schemes or identity based encryption and identity based signature schemes compared to Sign-then-Encrypt approach or other existing generic constructions which use encryption and signature schemes to construct signcryption schemes.

1.3 Organisation of the Thesis

Chapter 1 contains the introduction. In chapter 2, we provide the necessary preliminary material required in later chapters.

In chapter 3 [77], we provide a new security notion called Indistinguishability against Chosen Ciphertext Verification Attack (IND-CCVA). In existing security notions for public key encryption schemes, adversary may or may not be given access to a decryption oracle. The nature of this oracle is *computational*, i.e., for a given query on any ciphertext, it returns the message which is the decryption of the queried ciphertext. In IND-CCVA, adversary has access to a different oracle called Chosen Ciphertext Verification Oracle which is *decisional* in nature. Upon querying over any ciphertext, this oracle returns boolean value, i.e., 1 if ciphertext is valid, else 0.

Motivation of this notion comes from Bleichenbacher's [14] attack on PKCS #1 which was, at that time, termed as a chosen ciphertext attack. But our investigation shows that, in a stricter manner, this attack can be termed as a chosen ciphertext verification attack. Chapter 3 formalizes the notion of IND-CCVA and then shows that Cramer-Shoup light version [31] is an example of IND-CCVA secure encryption scheme.

Trivially, IND-CCVA secure public key encryption schemes are IND-CPA secure also and IND-CCA secure schemes are IND-CCVA secure also. We, in this chapter, investigated and showed the existence of IND-CCVA secure schemes which are not IND-CCA secure and then the existence of IND-CPA secure scheme which are not IND-CCVA secure and thus showed a gap between IND-CPA and IND-CCVA and then the same between IND-CCVA and IND-CCA.

The next two chapters contain the generic construction of signcryption schemes from encryption and signature schemes in identity based setting and traditional public key. In the year 1997, Yulian Zheng [94] proposed a new primitive *viz* signcryption in which confidentiality and authentication are achieved simultaneously at low communication and computational overhead. An-Dodis-Rabin [3] proposed a generic construction of signcryption schemes in public key setting using Commit then Encrypt and Sign paradigm. It was efficient in both signcrypt and designcrypt phases but it would not provide dM-IND-iCCA secure signcryption schemes. Lifting this paradigm in identity based setting would also result in IND-IBSC-gCCA secure signcryption schemes, a less secure than IND-IBSC-CCA secure signcryption schemes.

In an attempt to construct IND-IBSC-CCA secure signcryption schemes, chapter 4 [75, 76] provides two generic constructions of achieving IND-IBSC-CCA secure identity based signcryption schemes from IND-ID-CCA secure identity based encryption schemes. The difference between these constructions lies in the computational efficiency. The first construction is efficient in signcrypt phase in which encrypt and sign algorithms can be run in parallel. Whereas the second construction is efficient in both signcrypt and designcrypt phases due to possible parallelisation of encrypt and sign algorithms during signcrypt phase and that of

decrypt and verify algorithms during designcrypt phase. Both constructions use one IND-ID-CCA secure identity based encryption scheme and one SUF-ID-CMA secure identity based signature scheme. Where first construction uses three hash functions, second construction uses only two. Both constructions result in IND-IBSC-CCA and ESUF-IBSC-CMA secure signcryption schemes in the random oracle model.

Chapter 5 provides the generic constructions of achieving dM-IND-iCCA secure signcryption schemes from (a) OWE-CPA secure and (b) OWE-PCA secure encryption schemes. OWE-CPA is the minimum security required for any public key encryption scheme whereas dM-IND-iCCA is the highest known level of security for any signcryption scheme in public key setting. Another construction using relatively more secure public key encryption scheme than OWE-CPA, *viz* OWE-PCA secure, gives different but almost similar signcryption scheme which achieves the same level of security. Both these constructions are computationally efficient because encrypt and sign algorithms can be run in parallel during signcrypt phase as well as decrypt and verify algorithms can be run in parallel during designcrypt phase.

In chapter 6, we propose a new variant of Ring Signature called Ring Signature with Designated Verifier for Signer-Identity and provide one generic construction. Ring signature enables a user to sign a message so that a "ring" of possible signers (of which the user is a member) is identified, without revealing the actual signer. Our proposed variant additionally enables the signer to reveal its identity to a designated verifier at later stage. The whole algorithm can be divided into three phases - (a) Signer first generates a ring signature designated for a verifier on some chosen message and publishes them on public domain so that anybody can verify the ring signature, then (b) Signer then reveals a secret value *viz* trapdoor publicly and (c) Once trapdoor is published, only designated verifier may know the identity of actual signer. Such variants are useful in those scenarios where user (signer) may seek for reward at later stage.

This chapter describes the same variant of Ring Signature which was proposed in [84], however [84] lacks the proper formalization of the security notion of Ring Signature with Designated Verifier for Signer-Identity. This chapter formalizes the definition of the proposed variant and its security notions and moreover proposed a generic construction which is different from the construction proposed in [84]. The proposed construction achieves desired security properties in the random oracle model using an unconditionally anonymous and unforgeable ring signature, an identity based collision resistant (defined in chapter 2, subsection 2.9.2) and ANON-IBSC-CCA secure identity based signcryption schemes and two hash functions.

Chapter 2

Definitions and Preliminaries

2.1 Negligible Functions :

A function $f : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function if for every $c \in \mathbb{N}$, there exists $N_c \in \mathbb{N}$ such that for all $n > N_c$,

$$f(n) < \frac{1}{n^c}$$

For example, 2^{-n} and $n^{-log_2 n}$ are negligible functions. Functions which are not negligible are called non-negligible functions.

2.2 Hash Functions :

A hash function H is a function from $\{0,1\}^*$ (arbitrary finite length string) to $\{0,1\}^n$ (fixed length string), where $n \in \mathbb{Z}^+$. For cryptographic purpose, H should be easily computable.

2.2.1 Security Notions of Hash Functions

1. **Pre-image resistance** - A hash function H is called pre-image resistant if for almost all pre-specified outputs, it is computationally infeasible to find any input which maps to the specified output, i.e., for a given input $y \in \{0,1\}^n$, it should be difficult (computationally infeasible) to find $x \in \{0,1\}^*$ such that $H(x) = y$.

2. **Second Pre-image resistance** - A hash function H is called second pre-image resistant if it is computationally infeasible to find second input which maps to the same output, i.e., for a given input $x_1 \in \{0,1\}^*$, it should be difficult (computationally infeasible) to find second input $x_2 \in \{0,1\}^*$ and $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

3. **Collision resistance** - A hash function H is called collision resistant if it is computationally infeasible to two inputs which map to the same output, i.e., it should be difficult (computationally infeasible) to find two inputs x_1 and $x_2 \in \{0,1\}^*$ and $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

2.2.2 Random Oracle Model

Random oracle model tries to capture the notion of an ideal hash function. For an ideal hash function, the only way to determine the value of $H(x)$ is to actually calculate the value of H at x and it should be true even if many other values $(x_1, H(x_1))$, $(x_2, H(x_2))$, ... are known or computed. This model was proposed by Bellare and Rogaway [11] which gives a mathematical model to capture the notion of an ideal hash function. In this model, a hash function $H : \{0,1\}^* \to \{0,1\}^n$ is chosen at random from the set of all functions which map from $\{0,1\}^*$ to $\{0,1\}^n$ and then an oracle access is provided to compute the values of H. In other words, in this model, no formula or algorithm is provided to compute the values of function H; the only way to know is to query the oracle. It can be thought of as looking the value of $H(x)$ in a giant list of random numbers such that there is a random value of $H(x)$ for each x. Although there is no true random oracle in the existing life, we hope that a well designed function will behave like a random oracle model. This model provides a bridge between cryptographic theory and cryptographic practice. This paradigm yields protocols much more efficient than standard ones while retaining many of the advantages of the provable security.

2.3 Public Key Encryption Schemes :

A public key encryption scheme S_E is given by the following algorithms:

- **KG(1^λ):** A probabilistic polynomial time algorithm which takes security parameter 1^λ as input and outputs a public-private key pair (PK, SK).
- **ENC(m, PK):** A probabilistic polynomial time algorithm which takes a message m and public key PK as input and returns ciphertext $\mathcal{C}$.
- **DEC($\mathcal{C}, SK, PK$):** A deterministic polynomial time algorithm which takes ciphertext $\mathcal{C}$, secret key SK and public key PK as input and returns a message m if C is a valid ciphertext else $\perp$.

For consistency, it is required that for all $(PK, SK) \leftarrow \mathrm{KG}(1^\lambda)$ and all messages m, $m = \mathrm{DEC}(\mathrm{ENC}(m, PK), SK, PK)$.

The random string for the probabilistic encryption algorithm ENC can be generated internally or can be provided externally. If the random string $r = r_1||r_2|| \ldots ||r_k$ where $k \in \mathbb{Z}^+ \cup \{0\}$ (may be an empty string if $k = 0$) is provided externally for encryption algorithm, we denote it by ENC_r, else by simply ENC. It can be easily observed that for a given random string r and a message m, the encryption algorithm outputs a unique ciphertext c.

2.3.1 Security Notions of Encryption Schemes

A public key encryption scheme is said to be **OWE (One Way Encryption)** secure if no probabilistic polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage, where the advantage of $\mathcal{A}$ is defined as

$\mathcal{A}dv(\mathcal{A}) = \Pr[(PK, SK) \leftarrow \text{KG}(1^\lambda); \mathcal{C} \leftarrow \text{ENC}(m, PK); \mathcal{A}(\mathcal{C}, PK) = \text{DEC}(\mathcal{C}, SK, PK)]$.

OWE-PCA : An OWE secure encryption scheme S_E is said to be OWE-PCA secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to a plaintext checking oracle $\mathcal{O} = \{Plaintext\ Checking\}$.

- *Plaintext Checking*: Given a message-ciphertext pair, $(m, \mathcal{C})$, the oracle returns 1 if $m = DEC(\mathcal{C}, SK, PK)$ else 0.

$\underline{\textbf{Game}^{OWE-PCA}_{S_E,\mathcal{A}}(1^\lambda)}$

- $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
- $(st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK)$
- $\mathcal{C}_C \leftarrow \text{ENC}(m_C, PK)$
- $m' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_C, PK, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m_C = m']$.

Informally, no adversary can invert the ciphertext easily to get back the original message even if he/she has access to an oracle that verifies the validity of different message-ciphertext pairs. In the above game, adversary has access to this oracle before and after getting the challenged ciphertext. It should be noted that there is no restriction over the queried message-ciphertext pair, i.e., adversary can query over the message-challenged ciphertext pair also.

IND-CPA : A public key encryption scheme S_E is said to be **IND-CPA (indistinguishable against chosen plaintext attack)** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game.

$\underline{\textbf{Game}^{IND-CPA}_{S_E,\mathcal{A}}(1^\lambda)}$

- $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
- $(m_0, m_1, st) \leftarrow \mathcal{A}_1(PK)$; $|m_0| = |m_1|$
- $b \xleftarrow{R} \{0, 1\}$
- $\mathcal{C}_b \leftarrow \text{ENC}(m_b, PK)$
- $b' \leftarrow \mathcal{A}_2(C_b, PK, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$.

IND-CCA1/CCA2 : A public key encryption scheme S_E is said to be **IND-CCA1/CCA2 (indistinguishable against chosen ciphertext attack (/adaptive chosen ciphertext attack))** secure if no probabilistic polynomial time algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to a decryption oracle $\mathcal{O} = \{Decryption\}$.

- *Decryption*: Given a ciphertext $\mathcal{C}$, except the challenge ciphertext, the oracle returns $m/\perp \leftarrow \text{DEC}(\mathcal{C}, SK, PK)$.

$\textbf{Game}^{IND-CCA1}_{S_E,\mathcal{A}}(1^\lambda)$	$\textbf{Game}^{IND-CCA2}_{S_E,\mathcal{A}}(1^\lambda)$
• $(PK, SK) \leftarrow \text{KG}(1^\lambda)$	• $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK)$; $\lvert m_0\rvert = \lvert m_1\rvert$	• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK)$; $\lvert m_0\rvert = \lvert m_1\rvert$
• $b \xleftarrow{R} \{0,1\}$	• $b \xleftarrow{R} \{0,1\}$
• $\mathcal{C}_b \leftarrow \text{ENC}(m_b, PK)$	• $\mathcal{C}_b \leftarrow \text{ENC}(m_b, PK)$
• $b' \leftarrow \mathcal{A}_2(C_b, PK, st)$	• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C_b, PK, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$. It is easy to observe that in IND-CCA1 game, adversary $\mathcal{A}$ does not have access to decryption oracle once the challenged ciphertext $\mathcal{C}_b$ is obtained whereas in IND-CCA2, $\mathcal{A}$ has access to this oracle throughout the game resulting IND-CCA2 a stronger notion than IND-CCA1.

2.4 Public Key Signature Schemes :

A public key signature scheme S_S is given by the following algorithms:

- **KG(1^λ):** A probabilistic polynomial time algorithm which takes security parameter 1^λ as input and outputs a public-private key pair (PK, SK).
- **SIG(m, SK, PK):** A probabilistic polynomial time algorithm which takes a message m, a secret key SK, public key PK as input and outputs a signature σ.
- **VER(m, σ, PK):** A deterministic polynomial time algorithm which takes a message m, a signature σ, and public key PK as input and outputs true if σ is a valid signature on message m, else it returns false.

2.4.1 Security Notions of Signature Schemes

A signature scheme S_S is said to be **EUF/SUF-CMA (existentially/strongly unforgeable against chosen message attack)** secure if no probabilistic polynomial time algorithm has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to a signature oracle $\mathcal{O} = \{Signature\}$.

- *Signature*: Given a message m, the oracle returns $\sigma \leftarrow \text{SIG}(m, SK, PK)$ and adds (m, σ) to the list L.

$\underline{\textbf{Game}^{EUF-CMA}_{S_S,\mathcal{A}}(1^\lambda)}$	$\underline{\textbf{Game}^{SUF-CMA}_{S_S,\mathcal{A}}(1^\lambda)}$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $(PK, SK) \leftarrow \text{KG}(1^\lambda)$	• $(PK, SK) \leftarrow \text{KG}(1^\lambda)$
• $(m_C, \sigma_C) \leftarrow \mathcal{A}^{\mathcal{O}}(PK)$	• $(m_C, \sigma_C) \leftarrow \mathcal{A}^{\mathcal{O}}(PK)$
• $x \leftarrow \text{VER}(m_C, \sigma_C, PK)$	• $x \leftarrow \text{VER}(m_C, \sigma_C, PK)$

Advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[x = true \wedge (m_C, \sigma') \notin L]$. In EUF-CMA, σ' can be any string but in SUF-CMA $\sigma' = \sigma_C$.

2.5 Public Key Signcryption Schemes :

A public key signcryption scheme S_{SC} is given by the following algorithms:

- **Setup(1^λ):** A probabilistic polynomial time algorithm which takes a security parameter 1^λ as input and outputs the public parameters $Params$.
- **KG**$_{Rec}(Params)$**:** A probabilistic polynomial time algorithm which takes $Params$ as input and outputs the public-private (secret) receiver key pair (PK_{Rec}, SK_{Rec}).
- **KG**$_{Sen}(Params)$**:** A probabilistic polynomial time algorithm which takes $Params$ as input and outputs the public-private sender key pair (PK_{Sen}, SK_{Sen}).
- **SC**$(m, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$**:** A probabilistic polynomial time algorithm which takes a message m, a sender's secret key SK_{Sen}, a receiver's public key PK_{Rec}, sender's public key PK_{Sen} and public parameters $Params$ as input and returns ciphertext $\mathcal{C}$.
- **DSC**$(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$**:** A deterministic polynomial time algorithm which takes a ciphertext $\mathcal{C}$, a receiver's secret key SK_{Rec}, a receiver's public key PK_{Rec}, a sender's public key PK_{Sen} and public parameters $Params$ and returns either a message m if $\mathcal{C}$ is a valid ciphertext, else it returns $\perp$.

For consistency, it is required that for all $Params \leftarrow \text{Setup}(1^\lambda)$, for all identities of receiver Rec and sender Sen and for all messages m, if $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$, $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$, then
$m = \text{DSC}(\text{SC}(m, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params), SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$.

2.5.1 Security Notions of Signcryption Schemes

The security of signcryption schemes can be viewed in two settings-'two-user' and 'multi-user'. In two-user setting, there is only one sender and receiver whereas in multi-user setting, there can be more than one sender and receiver. In real scenario, adversary can sniff the channel and he/she can get ciphertexts corresponding to different senders and different receivers. This is the multi-user setting scenario. Then the question arises - 'why two-user setting?'. Although the two-user setting does not capture the real scenario, it may be helpful in building any signcryption scheme which is secure in the multi-user setting. The idea is to build a scheme which is secure in the two-user setting and then do some minor but proper modifications to make it secure in the multi-user setting. The general idea (which is **neither necessary nor sufficient**) to make a scheme secure in the multi-user setting from the two-user setting is:

1. During encryption, include the identity of the sender ID_{Sen} together with the encrypted message.

2. During signature generation, include the identity of the receiver ID_{Rec} together with the signed message.

3. At the receiver's end, output $\perp$, if any of expected sender's or receiver's identity does not match.

Moreover, one can consider two security models - (i) outsider (weaker) (ii) insider (stronger). In the insider security model, the sender or the receiver can act as an adversary but in the outsider security model, adversary can neither be a sender nor a receiver. From the model itself, it is clear that the insider security model is stronger than the outsider security model.

Security Notions in Two-User Setting

Now, we define the security notions in two-user insider and two-user outsider security model. In both models, the sender and the receiver will be fixed at the beginning of the game. For elaborate discussion on security for signcryption, one can refer to chapter 2 of [35].

Confidentiality: A signcryption scheme S_{SC} is said to be **T-IND-o/iCCA (indistinguishable against outsider/insider chosen ciphertext attack under two-user setting)** secure if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, the adversary has access to a designcryption oracle $\mathcal{O} = \{Designcryption\}$.

- $Designcryption$: Given a ciphertext $\mathcal{C}$, except the challenge ciphertext, the oracle returns $m/\ \perp \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$, where SK_{Rec}, PK_{Rec} and PK_{Sen} are generated in the beginning of the game and they are fixed.

$\mathbf{Game}^{T-IND-oCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{T-IND-iCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$; $\lvert m_0\rvert = \lvert m_1\rvert$	• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Sen}, Params)$; $\lvert m_0\rvert = \lvert m_1\rvert$
• $b \xleftarrow{R} \{0,1\}$	• $b \xleftarrow{R} \{0,1\}$
• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$
• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, Params, st)$	• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, SK_{Sen}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$

Unforgeability: A signcryption scheme S_{SC} is said to be **T-EUF-o/iCMA (existentially unforgeable against outsider/insider chosen message attack under two-user setting)** secure if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- $Signcryption$: Given a message m, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$ where SK_{Sen}, PK_{Rec} and PK_{Sen} are generated in the beginning of the game. The oracle then adds m to the list L.

$\mathbf{Game}^{T-EUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{T-EUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge m \notin L]$

A signcryption scheme S_{SC} is said to be **T-SUF-o/iCMA (strongly unforgeable against outsider/insider chosen message attack under two-user setting)** secure if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- *Signcryption* : Given a message m, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$ where SK_{Sen}, PK_{Rec} and PK_{Sen} are generated in the beginning of the game. The oracle then adds $(m, \mathcal{C})$ to the list L.

$\mathbf{Game}^{T-SUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{T-SUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $\mathcal{C} \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C} \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge (m, \mathcal{C}) \notin L]$

Security Notions in Multi-User Setting

Now, we define the security notions in multi-user setting. It can be further divided into two different settings - (a) fixed multi-user and (b) dynamic multi-user setting. In fixed multi-user setting, the receiver and sender are fixed at the beginning of the game whereas in dynamic multi-user setting only receiver (in confidentiality) or sender (in unforgeability) is fixed. Then what is difference between two-user and fixed multi-user setting? The difference lies at oracle query. In two-user setting, adversary is allowed to query only over fixed receiver's and sender's identities (fixed at the beginning of the game). Whereas in fixed multi-user setting, in the game of confidentiality, adversary can query the designcryption oracle over different ciphertexts and different sender's identities (except the challenge ciphertext and the fixed sender's identity) but fixed identity of receiver that is fixed at the beginning of the game. In the same way, in the game of unforgeability, adversary can query the signcryption oracle over different messages and different receiver's identities but fixed identity of sender that is fixed at the beginning of the game.

The difference in the two-user setting lies at the challenge phase. In the fixed multi-user, at the beginning of the game, the sender's and receiver's identities are fixed on which the attacker will mount an attack whereas in the dynamic multi-user, only one identity is fixed, namely, the receiver's identity in the game of confidentiality and the sender's identity in the game of unforgeability. Then, at the challenge phase, in the fixed multi-user model, the adversary will take those sender's and receiver's identities which are fixed at the beginning of the game. Whereas in the dynamic multi-user setting, in the game of confidentiality, the adversary chooses the sender's identity and similarly in the game of unforgeability, the receiver's identity is chosen at the challenge phase.

Now, we define the security notions in the fixed multi-user outsider/insider security model. In the fixed multi-user model, the public-private key pair of the receiver and the sender are generated at the beginning of the game. In the insider security model, the adversary knows the private key of the sender (in the game of confidentiality) or the receiver (in the game of unforgeability). For more discussions on security for signcryption in multi-user setting, one can refer to chapter 3 of [35].

Confidentiality: A signcryption scheme S_{SC} is said to be **fM-IND-o/iCCA (indistinguishable against outsider/insider chosen ciphertext attack under fixed multi-user model)** secure [69] if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a designcryption oracle $\mathcal{O} = \{Designcryption\}$.

- *Designcryption* : Given a ciphertext $\mathcal{C}$, the sender's identity Sen^*, the oracle returns $m/\perp \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen^*}, Params)$, where SK_{Rec} and PK_{Rec} are generated in the beginning of the game. Query over (C_b,Sen) are not allowed. Here C_b refers to the challenge ciphertext and Sen refers to the sender's identity generated at the beginning of the game.

$\textbf{Game}^{fM-IND-oCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\textbf{Game}^{fM-IND-iCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$; $\lvert m_0\rvert = \lvert m_b\rvert$	• $(m_0, m_1, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Sen}, Params)$ $\lvert m_0\rvert = \lvert m_b\rvert$
• $b \xleftarrow{R} \{0,1\}$	• $b \xleftarrow{R} \{0,1\}$
• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$
• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, Params, st)$	• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, SK_{Sen}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$

Unforgeability: A signcryption scheme S_{SC} is said to be **fM-EUF-o/iCMA (existentially unforgeable against outsider/insider chosen message attack under fixed multi-user model)** secure [69] if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- *Signcryption* : Given a message m and receiver's identity Rec^*, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec^*}, PK_{Sen}, Params)$ where PK_{Rec} and SK_{Sen} are generated in the beginning of the game. The oracle then adds m to the list L.

$\mathbf{Game}^{fM-EUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{fM-EUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge m \notin L]$

A signcryption scheme S_{SC} is said to be **fM-SUF-o/iCMA (strongly unforgeable against outsider/insider chosen message attack under fixed multi-user model)** secure if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- *Signcryption* : Given a message m and receiver's identity Rec^*, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec^*}, PK_{Sen}, Params)$ where PK_{Rec} and SK_{Sen} are generated in the beginning of the game. The oracle then adds $(m, \mathcal{C})$ to the list L.

$\mathbf{Game}^{fM-SUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{fM-SUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C} \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge (m, \mathcal{C}) \notin L]$

Similarly, we can define the security notions in the dynamic multi-user outsider/insider security model. In the dynamic multi-user model, public-private key pair of receiver (in the game of confidentiality) or sender (in the game of unforgeability) only is generated at the beginning of the game. In the insider security model, the adversary knows the private key of the sender (in the game of confidentiality) or the receiver (in the game of unforgeability).

Confidentiality: A signcryption scheme S_{SC} is said to be **dM-IND-o/iCCA (indistinguishable against outsider/insider chosen ciphertext attack under dynamic multi-user model)** secure [69] if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a designcryption oracle $\mathcal{O} = \{Designcryption\}$.

- *Designcryption* : Given a ciphertext $\mathcal{C}$, the sender's identity Sen^*, the oracle returns $m/\perp \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen^*}, Params)$, where SK_{Rec} and PK_{Rec} are generated in the beginning of the game. Query over (C_b,Sen) is not allowed. Here C_b refers to the challenge ciphertext and Sen refers to the sender's identity chosen by adversary at the challenge phase.

$\textbf{Game}^{dM-IND-oCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\textbf{Game}^{dM-IND-iCCA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$	• $(PK_{Rec}, SK_{Rec}) \leftarrow \text{KG}_{Rec}(Params)$
• $(m_0, m_1, Sen, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, Params)$; $\lvert m_0\rvert = \lvert m_1\rvert$	• $(m_0, m_1, Sen, st) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$; $\lvert m_0\rvert = \lvert m_1\rvert$
• $b \xleftarrow{R} \{0,1\}$	• $b \xleftarrow{R} \{0,1\}$
• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$	• $\mathcal{C}_b \leftarrow \text{SC}(m_b, SK_{Sen}, PK_{Rec}, PK_{Sen}, Params)$
• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, Params, st)$	• $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathcal{C}_b, PK_{Rec}, PK_{Sen}, SK_{Sen}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$

Unforgeability: A signcryption scheme S_{SC} is said to be **dM-EUF-o/iCMA (existentially unforgeable against outsider/insider chosen message attack under dynamic multi-user model)** secure [69] if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- *Signcryption* : Given a message m and receiver's identity Rec^*, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec^*}, PK_{Sen}, Params)$ where PK_{Sen} and SK_{Sen} are generated in the beginning of the game. The oracle then adds m to the list L.

$\mathbf{Game}^{dM-EUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{dM-EUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $(\mathcal{C}, Rec) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $(\mathcal{C}, Rec) \leftarrow \mathcal{A}_1^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge m \notin L]$

A signcryption scheme S_{SC} is said to be **dM-SUF-o/iCMA (strongly unforgeable against outsider/insider chosen message attack under dynamic multi-user model)** secure if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, the adversary $\mathcal{A}$ has access to a signcryption oracle $\mathcal{O} = \{Signcryption\}$.

- *Signcryption* : Given a message m and receiver's identity Rec^*, the oracle returns $\mathcal{C} \leftarrow \text{SC}(m, SK_{Sen}, PK_{Rec^*}, PK_{Sen}, Params)$ where PK_{Sen} and SK_{Sen} are generated in the beginning of the game. The oracle then adds $(m, \mathcal{C})$ to the list L.

$\mathbf{Game}^{dM-SUF-oCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$	$\mathbf{Game}^{dM-SUF-iCMA}_{S_{SC},\mathcal{A}}(1^\lambda)$
• $L \leftarrow \phi$	• $L \leftarrow \phi$
• $Params \leftarrow \text{Setup}(1^\lambda)$	• $Params \leftarrow \text{Setup}(1^\lambda)$
• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$	• $(PK_{Sen}, SK_{Sen}) \leftarrow \text{KG}_{Sen}(Params)$
• $(\mathcal{C}, Rec) \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, Params)$	• $(\mathcal{C}, Rec) \leftarrow \mathcal{A}^{\mathcal{O}}(PK_{Rec}, PK_{Sen}, SK_{Rec}, Params)$
• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$	• $m \leftarrow \text{DSC}(\mathcal{C}, SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge (m, \mathcal{C}) \notin L]$

2.6 Commitment Schemes :

Throughout this thesis, by a Commitment Scheme we mean a non-interactive Commitment Scheme. A Commitment Scheme consists of three algorithms:

- **Setup(1^λ):** A probabilistic polynomial time algorithm that takes security parameter 1^λ as input and outputs a commitment key CK.

- **Commit**(m, CK)**:** A probabilistic polynomial time algorithm which takes a message m, the commitment key CK and outputs a pair (c, d), where c is the commitment and d is the decommitment.
- **Open**$((c, d), CK)$**:** A deterministic polynomial time algorithm which takes commitment-decommitment pair (c, d) and the commitment key CK as input and returns m if (c, d) is a valid pair for m, else $\perp$.

For consistency, it is required that Open(Commit(m, CK),$CK) = m$ for all message $m \in \mathcal{M}$.

2.6.1 Security Notions of Commitment Schemes

Hiding Property : A commitment scheme S_{COMM} is said to have the hiding property if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game.

$\mathbf{Game}^{hide}_{S_{COMM},\mathcal{A}}(1^\lambda)$

- $CK \leftarrow \text{Setup}(1^\lambda)$
- $(m_0, m_1, st) \leftarrow \mathcal{A}_1(CK)$
- $b \overset{R}{\leftarrow} \{0, 1\}$
- $(c_b, d_b) \leftarrow \text{Commit}(m_b, CK)$
- $b' \leftarrow \mathcal{A}_2(c_b, CK, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$

Binding Property : A commitment scheme S_{COMM} is said to have binding property if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

$\mathbf{Game}^{bind}_{S_{COMM},\mathcal{A}}(1^\lambda)$

- $CK \leftarrow \text{Setup}(1^\lambda)$
- $((c, d, d')) \leftarrow \mathcal{A}(CK)$
- $m \leftarrow \text{Open}((c, d), CK)$
- $m' \leftarrow \text{Open}((c, d'), CK)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq m' \wedge (m, m') \neq \perp]$

Relaxed Binding Property : A commitment scheme S_{COMM} is said to have the relaxed binding property if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game.

$\mathbf{Game}^{r-bind}_{S_{COMM},\mathcal{A}}(1^\lambda)$

- $CK \leftarrow \text{Setup}(1^\lambda)$
- $(m, st) \leftarrow \mathcal{A}_1(CK)$
- $(c, d) \leftarrow \text{Commit}(m, CK)$
- $d' \leftarrow \mathcal{A}_2((c, d), CK, st)$
- $m' \leftarrow \text{Open}((c, d'), CK)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq m' \wedge (m, m') \neq \perp]$

2.7 Identity Based Encryption Schemes :

An Identity Based Encryption IBE scheme consists of four algorithms:

- **Setup**(1^λ) **:** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes security parameter 1^λ as input and outputs a master secret key MSK and public parameters $Params$.
- **KeyGen**$(ID, MSK, Params)$ **:** A probabilistic polynomial time algorithm run by the PKG which takes identity ID, master secret key MSK and public parameters $Params$ as input and outputs a secret key SK_{ID} associated to the identity ID.
- **Encrypt**$(m, ID_{Rec}, Params)$ **:** A probabilistic polynomial time algorithm that takes a message m, the recipient's identity ID_{Rec} and public parameters $Params$ as input and outputs a ciphertext $\mathcal{C}$.
- **Decrypt**$(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, Params)$ **:** A deterministic polynomial time algorithm that takes a ciphertext $\mathcal{C}$, the recipient's identity ID_{Rec}, the recipient's secret key $SK_{ID_{Rec}}$ and public parameters $Params$ as input and outputs m if $\mathcal{C}$ is a valid ciphertext of message m else $\perp$.

For consistency, it is required that for all $(MSK, Params) \leftarrow \text{Setup}(1^\lambda)$, all messages m and for all receiver's identity ID_{Rec},

$$\text{Decrypt}(\text{Encrypt}(m, ID_{Rec}, Params), ID_{Rec}, SK_{ID_{Rec}}, Params) = m.$$

2.7.1 Security Notions of IBE Schemes

An IBE scheme S_{IBE} is said to be **IND-ID-CCA (indistinguishable against adaptively chosen-ciphertext attack under identity based setting)** secure if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to two oracles, $\mathcal{O}_1 = \{KeyGeneration\}$ and $\mathcal{O}_2 = \{Decryption\}$.

- $KeyGeneration$: Given an identity ID, except the challenge identity, the oracle returns $SK_{ID} \leftarrow \text{KeyGen}(ID, MSK, Params)$.
- $Decryption$: Given a ciphertext $\mathcal{C}$ and the receiver's identity ID_{Rec^*}, the oracle returns $m/\perp \leftarrow \text{Decrypt}(\mathcal{C}, SK_{ID_{Rec^*}}, ID_{Rec^*}, Params)$. Query over $(\mathcal{C}_b, ID_{Rec})$ is not allowed where $\mathcal{C}_b$ is the challenge ciphertext and ID_{Rec} is the identity chosen by the adversary at the challenge phase.

$\mathbf{Game}^{IND-ID-CCA}_{S_{IBE},\mathcal{A}}(1^\lambda)$

- $(MSK, Params) \leftarrow \text{Setup}(1^\lambda)$
- $(m_0, m_1, ID_{Rec}, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2}(Params)$; $|m_0| = |m_1|$
- $b \overset{R}{\leftarrow} \{0, 1\}$
- $\mathcal{C}_b \leftarrow \text{Encrypt}(m_b, ID_{Rec}, Params)$
- $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2}(\mathcal{C}_b, ID_{Rec}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$

We define an IBE scheme S_{IBE} to be **IND-ID-gCCA** secure if no probabilistic polynomial time adversary has a non-negligible advantage in the game which is the same as that of IND-ID-CCA (described above) except that while querying the $Decryption$ oracle $\mathcal{O}_2$, the adversary is not allowed to query on ciphertext $\mathcal{C}$ with the same identity ID_R such that $Decrypt(\mathcal{C}_b, ID_R, SK_R, Params) = Decrypt(\mathcal{C}, ID_R, SK_R, Params)$.

Note that in the game of **IND-ID-gCCA**, ID_R may or may not be equal to ID_{Rec}. In response to queries on the $Decryption$ oracle, the restriction is on all the decryption queries on ciphertext $\mathcal{C}$ with ID_R which gives the same message as queried on the challenge ciphertext $\mathcal{C}_b$ with ID_R. It is straightforward that IND-ID-CCA secure IBE schemes will be IND-ID-gCCA secure also. If the IBE scheme is malleable in the sense that the ciphertext $\mathcal{C}$ can be modified to another ciphertext $\mathcal{C}'$ such that $\mathcal{C}$ and $\mathcal{C}'$ yields different messages, say m and m', upon decryption with the relation $\mathcal{R}$ known between the messages (i.e. $\mathcal{R}(m, m')$ is known), then that IBE scheme cannot be IND-ID-gCCA secure.

2.8 Identity Based Signature Schemes :

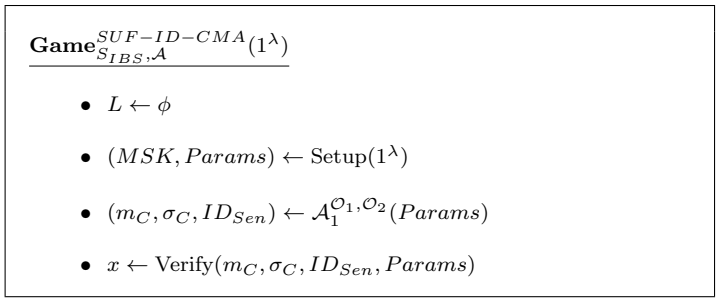
An Identity Based Signature (IBS) scheme consists of four algorithms:

- **Setup(1^λ) :** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes a security parameter 1^λ as input and outputs a master secret key MSK and public parameters $Params$.
- **KeyGen($ID, MSK, Params$) :** A probabilistic polynomial time algorithm run by PKG which takes identity ID, master secret key MSK and public parameters $Params$ as input and outputs a secret key SK_{ID} associated to the identity ID.
- **Sign($m, SK_{ID_{Sen}}, ID_{Sen}, Params$) :** A probabilistic polynomial time algorithm that takes a message m, the sender's secret key $SK_{ID_{sen}}$, the sender's identity ID_{Sen} and public parameters $Params$ as input and outputs a signature σ.
- **Verify($m, \sigma, ID_{Sen}, Params$) :** A deterministic polynomial time algorithm that takes a message m, a signature σ, the sender's identity ID_{Sen} and public parameters $Params$ as input and outputs true if σ is a valid signature of the message m, else outputs false.

2.8.1 Security Notions of IBS Schemes

An IBS S_{IBS} scheme is said to be **SUF-ID-CMA (strongly unforgeable against chosen message attack under identity based setting)** if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to two oracles, $\mathcal{O}_1 = \{KeyGeneration\}$ and $\mathcal{O}_2 = \{Signature\}$.

- $KeyGeneration$: Given an identity ID, except the challenge identity, the oracle returns $SK_{ID} \leftarrow \text{KeyGen}(ID, MSK, Params)$.
- $Signature$: Given a message m and the sender's identity ID_{Sen^*}, the oracle returns $\sigma \leftarrow \text{Sign}(m, SK_{ID_{Sen^*}}, ID_{Sen^*}, Params)$ and adds (m, σ, ID_{Sen^*}) to the list L.



The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[x = \text{true} \wedge (m_C, \sigma_C, ID_{Sen}) \notin L]$

2.9 Identity Based Signcryption Schemes :

An Identity Based Signcryption IBSC scheme consists of four algorithms (chapter 10 of [35]; chapter 2, page 151 of [63]):

- **Setup**(1^λ) **:** A probabilistic polynomial time algorithm run by a private key generator (PKG) that takes security parameter 1^λ as input and outputs a master secret key MSK and public parameters $Params$.

- **KeyGen**$(MSK, Params)$ **:** A probabilistic polynomial time algorithm run by the PKG which takes the master secret key MSK, an identity ID and public parameters $Params$ as input and outputs a secret key SK_{ID} associated to the identity ID.

- **Signcrypt**$(m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$ **:** A probabilistic polynomial time algorithm that takes a message m, the sender's secret key $SK_{ID_{Sen}}$, the receiver's identity ID_{Rec}, the sender's identity ID_{Sen} and public parameters $Params$ as input and outputs a ciphertext $\mathcal{C}$.

- **Designcrypt**$(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$ **:** A deterministic polynomial time algorithm that takes a ciphertext $\mathcal{C}$, the receiver's secret key $SK_{ID_{Rec}}$, the receiver's identity ID_{Rec}, the sender's identity ID_{Sen} and public parameters $Params$ as input and outputs either a tuple (m, s) if $\mathcal{C}$ is a valid ciphertext of m, else an error symbol $\perp$. Here, s is an additional information that allows the receiver to convince a third party that the message actually emanated from the sender.

For consistency, it is required that if

$$\mathcal{C} = Signcrypt(m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$$

then the output of the $Designcrypt(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$ should be (m, s) where s is an additional information that allows the receiver Rec to convince a third party that the message m actually emanated from the sender Sen.

2.9.1 Security Notions of IBSC Schemes

Confidentiality

An IBSC is said to be **IND-IBSC-CCA (indistinguishable against chosen ciphertext attack under identity based setting)** (chapter 10 of [35]; chapter 2, page 152 of [63]) if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to three oracles $\mathcal{O}_1 = \{KeyGeneration\}$, $\mathcal{O}_2 = \{Signcryption\}$ and $\mathcal{O}_3 = \{Designcryption\}$.

- $KeyGeneration$: Given an identity ID, except the challenge receiver's identity, the oracle returns $SK_{ID} \leftarrow \text{KeyGen}(ID, MSK, Params)$.

- $Signcryption$: Given a message m, the sender's identity ID_{Sen^*} and the receiver's identity ID_{Rec^*}, the oracle returns $\mathcal{C} \leftarrow \text{Signcrypt}(m, SK_{ID_{Sen^*}}, ID_{Sen^*}, ID_{Rec^*}, Params)$.

- *Designcryption* : Given a ciphertext $\mathcal{C}$, the receiver's identity ID_{Rec^*} and the sender's identity ID_{Sen^*}, the oracle returns $(m, s) \leftarrow Designcrypt(\mathcal{C}, SK_{ID_{Rec^*}}, ID_{Rec^*}, ID_{Sen^*}, Params)$. Query over $(\mathcal{C}_b, ID_{Rec}, ID_{Sen})$ is not allowed, where $\mathcal{C}_b$ is the challenge ciphertext, ID_{Rec} and ID_{Sen} are the challenge identity of receiver and sender respectively.

$\mathbf{Game}^{IND-IBSC-CCA}_{S_{IBSC},\mathcal{A}}(1^\lambda)$

- $(MSK, Params) \leftarrow \text{Setup}(1^\lambda)$
- $(m_0, m_1, ID_{Rec}, ID_{Sen}, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(Params)$; $|m_0| = |m_1|$
- $b \overset{R}{\leftarrow} \{0, 1\}$
- $\mathcal{C}_b \leftarrow \text{Signcrypt}(m_b, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$
- $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathcal{C}_b, ID_{Rec}, ID_{Sen}, SK_{ID_{Sen}}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b' = b] - \frac{1}{2}|$.

We define an IBSC scheme to be **IND-IBSC-gCCA** secure if no probabilistic polynomial time adversary has a non-negligible advantage in the game which is the same as that of IND-IBSC-CCA except that while querying the *Designcryption* oracle, the adversary is not allowed to query on ciphertext $\mathcal{C}$ on the same receiver's identity ID_{Rec} and some other sender's identity ID_{Sen^*} (may be different from ID_{Sen}) such that the first string of the output of the $Designcryption(\mathcal{C}_b, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$, i.e, message m_b and the first string of the output of the $Designcryption(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen^*}, Params)$, i.e., message m are same.

An IBSC is said to be **ANON-IBSC-CCA (anonymous against chosen ciphertext attack under identity based setting)** (chapter 10, page 210 of [35]; chapter 2, page 156 of [63]) if no probabilistic polynomial time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to three oracles $\mathcal{O}_1 = \{KeyGeneration\}$, $\mathcal{O}_2 = \{Signcryption\}$ and $\mathcal{O}_3 = \{Designcryption\}$.

- *KeyGeneration* : Given an identity ID, except the challenge receiver's identities, the oracle returns $SK_{ID} \leftarrow \text{KeyGen}(ID, MSK, Params)$.
- *Signcryption* : Given a message m, the sender's identity ID_{Sen^*} and the receiver's identity ID_{Rec^*}, the oracle returns $\mathcal{C} \leftarrow \text{Signcrypt}(m, SK_{ID_{Sen^*}}, ID_{Sen^*}, ID_{Rec^*}, Params)$.
- *Designcryption* : Given a ciphertext $\mathcal{C}$, the receiver's identity ID_{Rec^*} and the sender's identity ID_{Sen^*}, the oracle returns $(m, s) \leftarrow Designcrypt(\mathcal{C}, SK_{ID_{Rec^*}}, ID_{Rec^*}, ID_{Sen^*}, Params)$. Queries over $(\mathcal{C}_{b,\hat{b}}, ID_{Rec_0}, ID_{Sen})$ and $(\mathcal{C}_{b,\hat{b}}, ID_{Rec_0}, ID_{Sen})$ are not allowed, where $\mathcal{C}_{b,\hat{b}}$ is the challenge ciphertext, ID_{Rec_0}, ID_{Rec_1} are challenge identities of receivers and ID_{Sen} is any identity of sender.

$\mathbf{Game}^{ANON-IBSC-CCA}_{S_{IBSC},\mathcal{A}}(1^\lambda)$

- $(MSK, Params) \leftarrow \text{Setup}(1^\lambda)$
- $(m, ID_{Rec_0}, ID_{Rec_1}, ID_{Sen_0}, ID_{Sen_1}, st) \leftarrow \mathcal{A}_1^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(Params)$
- $b, \hat{b} \xleftarrow{R} \{0,1\}$
- $\mathcal{C}_{b,\hat{b}} \leftarrow \text{Signcrypt}(m, SK_{ID_{Sen_b}}, ID_{Rec_{\hat{b}}}, ID_{Sen_b}, Params)$
- $(b', \hat{b}') \leftarrow \mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(\mathcal{C}_{b,\hat{b}}, ID_{Rec_0}, ID_{Rec_1}, ID_{Sen_0}, ID_{Sen_1}, SK_{ID_{Sen_0}}, SK_{ID_{Sen_1}}, Params, st)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = |\Pr[(b', \hat{b}') = (b, \hat{b})] - \frac{1}{4}|$.

Note that in the game of ANON-IBSC-CCA, the ciphertext conveys no information either about the sender or the receiver. This game is same as IND-IBSC-CCA except this fact that challenge is on identities instead of the message (see chapter 10, page 210 of [35]). It is obvious from the definition that even if the ciphertext leaks information about the message, the security of ANON-IBSC-CCA may be satisfied unless it is disproved theoretically.

Unforgeability

An IBSC is said to be **ESUF-IBSC-CMA (signature unforgeable against chosen message attack under identity based setting)** (chapter 10, page 207 of [35]) if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to three oracles $\mathcal{O}_1 = \{KeyGeneration\}$, $\mathcal{O}_2 = \{Signcryption\}$ and $\mathcal{O}_3 = \{Designcryption\}$.

- $KeyGeneration$: Given an identity ID, except the challenge sender's identity, the oracle returns $SK_{ID} \leftarrow \text{KeyGen}(ID, MSK, Params)$.
- $Signcryption$: Given a message m, the sender's identity ID_{Sen^*} and the receiver's identity ID_{Rec^*}, the oracle returns $\mathcal{C} \leftarrow \text{Signcrypt}(m, SK_{ID_{Sen^*}}, ID_{Sen^*}, ID_{Rec^*}, Params)$. Let $(m, s) \leftarrow Designcrypt(\mathcal{C}, SK_{ID_{Rec^*}}, ID_{Rec^*}, ID_{Sen^*}, Params)$. Oracle then adds (m, s, ID_{Sen^*}) to the list L.
- $Designcryption$: Given a ciphertext $\mathcal{C}$, the receiver's identity ID_{Rec^*} and the sender's identity ID_{Sen^*}, the oracle returns $(m, s) \leftarrow Designcrypt(\mathcal{C}, SK_{ID_{Rec^*}}, ID_{Rec^*}, ID_{Sen^*}, Params)$.

$\mathbf{Game}^{ESUF-IBSC-CMA}_{S_{IBSC},\mathcal{A}}(1^\lambda)$

- $L \leftarrow \phi$
- $(MSK, Params) \leftarrow \text{Setup}(1^\lambda)$
- $(\mathcal{C}_C, ID_{Rec}, ID_{Sen}) \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}(Params)$
- $(m, s) \leftarrow \text{Designcrypt}(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$

The advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[m \neq \perp \wedge (m, s, ID_{Sen}) \notin L]$

2.9.2 Identity Collision Resistant Signcryption Schemes

We define an IBSC scheme to be *identity collision-resistant* (defined for this thesis; does not exist in the literature) if with negligible probability there exists two different sets of identities $\{ID_{Rec}, ID_{Sen}\} \neq \{ID_{Rec'}, ID_{Sen'}\}$ such that Signcrypt$(m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$ = Signcrypt$(m, SK_{ID_{Sen'}}, ID_{Rec'}, ID_{Sen'}, Params)$ for randomly chosen message m and all random coins used in the algorithm 'Signcrypt'.

An IND-IBSC-CCA secure scheme naturally satisfies this property as otherwise in the game of IND-IBSC-CCA, given the challenge ciphertext $\mathcal{C}_b$ on identities ID_{Rec} and ID_{Sen}, an adversary $\mathcal{A}$ can query on the ciphertext $\mathcal{C}_b$ with a different set of identities $\{ID_{Rec'}, ID_{Sen'}\} \neq \{ID_{Rec}, ID_{Sen}\}$ and will get the correct message m_b with non-negligible probability. Moreover, it can also be seen that the converse is not true and hence the set of identity collision resistant IBSC schemes is strictly the superset of IND-IBSC-CCA secure schemes. This property has been used in the construction of one RS-DVSI scheme in the chapter 6.

2.10 Ring Signature Schemes

A Ring Signature scheme S_R is given by the following algorithms:

- **Setup**(1^λ)**:** A probabilistic polynomial time algorithm which takes a security parameter 1^λ as an input and returns public parameters $Params$.
- **KeyGen**$(ID_u, Params)$**:** A probabilistic polynomial time algorithm which takes an identity ID_u corresponding to user u and public parameters $Params$ as input and returns a public key PK_u and a private (secret) key SK_u corresponding to the user u.
- **RSign**$(m, \mathcal{R}, SK_u, Params)$**:** A probabilistic polynomial time algorithm which takes a message m, a ring $\mathcal{R} = \cup_{i=1}^{n}\{PK_{u_i}\}$, a secret key corresponding to some user u ($PK_u \in \mathcal{R}$) and public parameters $Params$ as input and returns a signature σ on the message m corresponding to the ring $\mathcal{R}$.

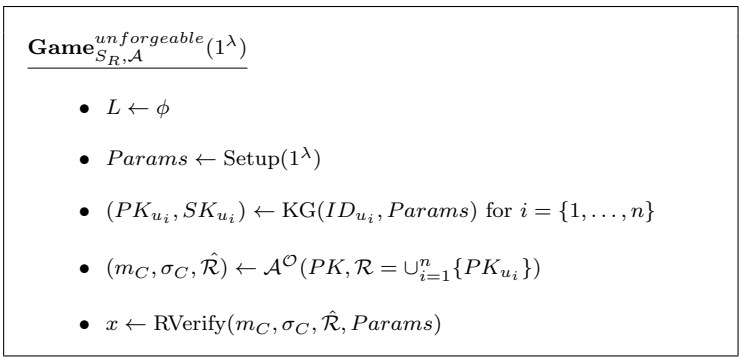
- **RVerify**$(m, \sigma, \mathcal{R}, Params)$**:** A deterministic polynomial time algorithm which takes a message m, a signature σ, a ring $\mathcal{R}$ and public parameters $Params$ as input and returns *true* if σ is a valid signature on the message m corresponding to the ring $\mathcal{R}$ else it returns *false*.

2.10.1 Security Notions of Ring Signature Schemes

Unconditional Anonymity: The adversary with unbounded computation power is given a signature computed by a randomly-chosen signer from theh ring $\mathcal{R}$, with the requirement that the adversary should be unable to guess the actual signer with the probability better than $\frac{1}{|\mathcal{R}|} + \epsilon$, where ϵ is a negligible function of the security parameter.

A ring signature scheme S_R is said to be **unforgeable** if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game. In this game, $\mathcal{A}$ has access to an oracle $\mathcal{O} = \{OSign\}$.

- $OSign$: Given a message m, a ring $\mathcal{R}' \subseteq \mathcal{R} = \cup_{i=1}^{n}\{PK_{u_i}\}$ ($\mathcal{R}$ is generated at the beginning of the game) and signer u_s, the oracle returns a signature σ on meassage m signed by the signer u_s corresponding to ring $\mathcal{R}'$. Oracle then adds $(m, \mathcal{R}')$ to the list L.



Advantage of $\mathcal{A}$ is defined as $\mathcal{A}dv(\mathcal{A}) = \Pr[x = true \wedge (m_C, \mathcal{R}') \notin L \wedge \hat{\mathcal{R}} \subseteq \mathcal{R}]$

Chapter 3

Relaxing IND-CCA: Indistinguishability against Chosen Ciphertext *Verification* Attack

The definition of IND-CCA security model for public key encryption allows an adversary to obtain (adaptively) decryption of ciphertexts of its choice. That is, the adversary is given oracle access to the decryption function corresponding to the decryption key in use. The adversary may make queries that do not correspond to a *valid* ciphertext, and the answer will be accordingly (i.e., a special "failure" symbol).

In this chapter, we investigate the case where we restrict the oracle to only determine if the query made is a valid ciphertext or not. That is, the oracle will output 1 if the query string is a valid ciphertext (do not output the corresponding plaintext) and output 0 otherwise. We call this oracle as "*ciphertext verification oracle*" and the corresponding security model as indistinguishability against *chosen ciphertext verification attack* (IND-CCVA). We point out that this seemingly weaker security model is meaningful, clear and useful to the extent where we motivate that certain cryptographic functionalities can be achieved by ensuring the IND-CCVA security where as IND-CPA is not sufficient and IND-CCA provides more than necessary. We support our claim by providing nontrivial construction (existing/new) of:

- public key encryption schemes that are IND-CCVA secure but not IND-CCA secure,
- public key encryption schemes that are IND-CPA secure but not IND-CCVA secure.
- public key encryption schemes that are IND-CCA1 secure but not IND-CCVA secure.

Our discoveries are another manifestation of the subtleties that make the study of security notions for public key encryption schemes so attractive and are important towards achieving the definitional clarity of the target security.

3.1 Introduction

The IND-CCA security (security against adaptive chosen ciphertext attacks [73, 80, 9, 38, 86]) is nowadays considered the *de facto* level of security required for public key encryption schemes used in practice. Some approaches are known for constructing encryption schemes that meet this notion of security. But, in practice, there are certain cryptographic functionalities for which the security requirement is apparently **less stronger** than IND-CCA. There had been wide variety of research to quantify the gap between the IND-CPA and IND-CCA security (see [30, 53, 52]).

The most common threat to IND-CCA security is that of a query on a malformed ciphertext causing the decryption oracle to leak damaging information, either about the private key, or about the plaintext. Understanding, the explicit behaviour of the decryption oracle could be the keypoint. In the IND-CCA model, the decryption oracle provides decryption of the ciphertexts of our choice. In this work, we limit the output of the decryption oracle: it will only verify whether or not a query string is a valid ciphertext or not. Consider a setting where a server has the secret key, receives and decrypts ciphertext and sends to the client an accept/reject message depending on whether the ciphertext was valid or not. This setting actually occurs in real life, some schemes can be broken in this setting, and Bleichenbacher's [14] attack is an example of this.

The security for public key encryption was first formally defined by Goldwasser and Micali [49]. Their notion of *semantic security*, roughly speaking, requires that observation of a ciphertext does not enable an adversary to compute anything about the underlying plaintext message that it could not have computed on its own (i.e., prior to observing the ciphertext). Goldwasser and Micali (see also [46, 47]) proved that semantic security is equivalent to the notion of *indistinguishability* that requires (roughly) the following: given a public key pk, a ciphertext C, and two possible plaintexts m_0, m_1, it is infeasible to determine if C is an encryption of m_0 or an encryption of m_1. We refer to these notions using the commonly accepted term "IND-CPA" security.

IND-CPA security does not guarantee any security against *chosen ciphertext* attacks by which an adversary may obtain decryption of ciphertexts of its choice. Indistinguishability based definitions appropriate for this setting were given by Naor and Yung [73] and Rackoff and Simon [80]. Naor and Yung consider *non-adaptive* chosen ciphertext attack in which the adversary may request decryptions only *before* it obtains the challenge ciphertext. Rackoff and Simon define the stronger notion of security against *adaptive* chosen ciphertext attacks whereby the adversary may request decryptions even after seeing the challenge ciphertext, under the natural limitation that the adversary may not request decryption of the challenge ciphertext itself. We will refer to the later notion as "IND-CCA" security. Lots of research have been done in this direction (see [89] and references there in). Loffus et al. [68] have studied IND-CCA and showed the importance of CCA-like notions in the security of cloud computing. Recently, in [90], CCA has been extended where adversary can not only exploit the decryption oracle queries but also the intermediate calculations stored in hardware (especially RAM). This new notion, where decryption oracle is referred as glass box decryption

oracle, is known as Glass-Box-CCA.

In the literature the paradigms that construct IND-CCA secure cryptosystems are few in number. Among them, the paradigms introduced by Naor and Yung in [73] and by Cramer and Shoup in [31, 32] are very famous. The proofs of well-formedness of ciphertexts have been shown to underlie the constructions that were instantiated by both of the above paradigms. Informally it speaks of a validity check step for ciphertexts in the decryption algorithm. Infact, Elkind and Sahai have observed [40] that both the above approaches for constructing CCA-secure encryption schemes can be viewed as special cases of a single paradigm. In this paradigm one starts with a CPA-secure cryptosystem in which certain ill-formed ciphertexts are indistinguishable from honestly-generated ciphertexts. A CCA-secure cryptosystem is then obtained by having the sender honestly generate a ciphertext using the underlying CPA-secure scheme, and then append a proof of well-formedness (satisfying certain criteria) to this ciphertext. Thus having a validity check seems a sufficient condition to achieve IND-CCA security and became a common practice until Bleichenbacher's [14] attack showed that this is not the case; the attack broke the IND-CCA security of the underlying scheme using a oracle that confirms just the validity of the ciphertext. Thus for the class of public key encryption schemes with validity check in the decryption could give rise to a meaningful security model (less stronger than IND-CCA) where the adversary has access to a oracle of the above nature. We name this security model as IND-CCVA.

In this chapter, we search for IND-CCVA secure public key encryption schemes. Beside its theoretical importance, there are some practical benefits as well. For example, consider the scenario where one has to pick a encryption scheme between $\mathcal{E}_1$ and $\mathcal{E}_2$, where both are IND-CPA secure but not IND-CCA and both of them are efficient. Suppose $\mathcal{E}_2$ differs with $\mathcal{E}_1$ by having a validity checking step in its decryption algorithm. In this case one may tend to prefer $\mathcal{E}_2$ over $\mathcal{E}_1$, but our findings will show that this may not be a wiser decision always. In the definition of IND-CCA security model for public key encryption, the adversary is given oracle access to the decryption function corresponding to the decryption key in use. The adversary may make queries that do not correspond to a *valid* ciphertext, and the answer will be accordingly (i.e., a special "failure" symbol). In this chapter, we investigate the case where we restrict the oracle to only verify if the query made is a valid ciphertext or not. That is, the oracle will output 1 (not the corresponding plaintext) if the query string is a valid ciphertext and output 0 otherwise. We will denote this oracle by the name "*ciphertext verification oracle*" and the corresponding security model by the name Indistinguishability against *chosen ciphertext verification attack* (IND-CCVA). We point out that this seemingly weaker security model is meaningful, clear and useful to the extent where we observe that certain cryptographic functionalities can be achieved by ensuring the IND-CCVA security where IND-CPA is not sufficient and IND-CCA provides more than necessary. We further support our claim by providing generic construction of:

- public key encryption schemes that are IND-CCVA secure but not IND-CCA secure,
- public key encryption schemes that are IND-CPA secure but not IND-CCVA secure,
- public key encryption schemes that are IND-CCA1 secure but not IND-CCVA secure.

3.2 IND-CCVA: Indistinguishability against Chosen Ciphertext Verification Attack

We now present a formal definition of security against chosen ciphertext verification attacks. This is a weaker form of attack when compared to a full CCA attack: the adversary has access to an oracle which is weaker than a decryption oracle. We name this oracle as ciphertext verification oracle and denoted it by $\mathcal{O}_{CV}$. The oracle is described as follows:

$$\mathcal{O}_{CV} : \{0,1\}^* \rightarrow \{0,1\}$$

The output is 1 if and only if the input string is a *valid ciphertext*. We now describe this new attack model formally as follows. For a public key encryption scheme $\prod$ and an adversary $\mathcal{A}$, consider the following experiment:

The IND-CCVA Experiment

- $\mathrm{KG}(1^\lambda)$ is run to obtain keys (PK, SK).
- Beside the public key PK, the adversary $\mathcal{A}$ is given access to the ciphertext verification oracle $\mathcal{O}_{CV}$.
- The adversary outputs a pair of messages m_0, m_1 of the same length from the message space.
- A random bit $b \leftarrow \{0,1\}$ is chosen at random, and then a ciphertext $\mathcal{C}_b \leftarrow \mathrm{ENC}(m_b, PK)$ is computed and given to $\mathcal{A}$.
- $\mathcal{A}$ continues to interact with $\mathcal{O}_{CV}$.
- Finally, $\mathcal{A}$ outputs a bit b'.

We define the advantage of $\mathcal{A}$ in the IND-CCVA experiment as follows:

$$\mathcal{A}dv(\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}| \tag{3.1}$$

Note that our oracle may become constant (always output 1) for certain class of public key encryption schemes. Let $\prod$ be a public key encryption scheme with $\mathbf{K}$ as key space, $\mathbf{M}$ as message space, and $\mathbf{C}$ as ciphertext space. In general, we have

$$\cup_{k \in \mathbf{K}} \mathrm{ENC}(\mathbf{M}) \subsetneq \mathbf{C}.$$

The equality between $\cup_{k \in \mathbf{K}} \mathrm{ENC}(\mathbf{M})$ and $\mathbf{C}$ means that any element from $\mathbf{C}$ is a valid ciphertext (encryption of some message under some key). Thus in this case, the verification oracle will always output 1 for any random query from $\mathbf{C}$. This will imply that the IND-CPA and IND-CCVA security are both equivalent for such public key encryption schemes (as oracle is of no use), for example, ElGamal.

In this chapter we consider public keys encryption schemes with $\cup_{k\in\mathbf{K}}\text{ENC}(\mathbf{M}) \subsetneq \mathbf{C}$. Infact, achieving IND-CCA security requires this kind of setup in general.

Remark: One may note that the security model of IND-CCVA immediately confirms the following:

- IND-CCA implies IND-CCVA and
- IND-CCVA implies IND-CPA

3.3 The Separating Scheme: IND-CCVA secure but not IND-CCA secure

In this section we describe a public key encryption scheme which was originally proposed by Cramer and Shoup [31] as the light version of their main scheme (the first practical IND-CCA secure scheme). The scheme was shown to be IND-CPA secure by Cramer and Shoup and not IND-CCA secure. We observed that this scheme is in-fact IND-CCVA secure, thus settling the claim of this section.

3.3.1 Cramer-Shoup light version

- **KG(1^λ):** The key generation algorithm runs as follows.
  - Choose a group G of prime order p, where $2^{\lambda-1} < p < 2^\lambda$
  - Choose $g_1, g_2 \xleftarrow{\mathcal{R}} G$ and $x_1, x_2, z \in \mathbb{Z}_p$.
  - Compute $c = g_1^{x_1} g_2^{x_2}$ and $h = g_1^z$.
  - The public key, PK, for this scheme is tuple (g_1, g_2, c, h), with corresponding secret key, SK, is (x_1, x_2, z).
  - message space $= G$.
- **ENC(m, PK):** To encrypt a message $m \in G$, the encryption algorithm runs as follows.
  - Choose $r \xleftarrow{\mathcal{R}} \mathbb{Z}_p$.
  - Compute $u_1 = g_1^r,\ u_2 = g_2^r,\ e = h^r m,\ v = c^r$.
  - The ciphertext, $\mathcal{C}$, is (u_1, u_2, e, v).
- **DEC($\mathcal{C}, SK, PK$):** Decryption works in the following way: given the ciphertext (u_1, u_2, e, v) and secret key (x_1, x_2, z),
  - It first tests if $u_1^{x_1} u_2^{x_2} \stackrel{?}{=} v$.
  - If this condition does not hold, the decryption algorithm outputs $\perp$; otherwise, it outputs
$$m = \frac{e}{u_1^z}.$$

Correctness. If (u_1, u_2, e, v) is a valid ciphertext, then we have:

$$u_1^{x_1} u_2^{x_2} = g_1^{r x_1} g_2^{r x_2} = g_1^{x_1} g_2^{x_2 r} = c^r = v \text{ and}$$

$$\frac{e}{u_1^z} = \frac{h^r m}{g_1^{rz}} = \frac{g_1^{zr} m}{g_1^{rz}} = \frac{g_1^{rz} m}{g_1^{rz}} = m.$$

3.3.2 IND-CCVA Security

We show that this scheme is IND-CCVA secure based on the hardness of the Decisional Diffie-Hellman (DDH) problem in G.

DDH problem can be formulated as follows. Let $\mathcal{D}$ be an algorithm that takes triples of group elements as input and outputs a bit. The DDH-advantage of $\mathcal{D}$ is defined as

$$\left| \Pr[x, y \xleftarrow{\mathcal{R}} \mathbb{Z}_p : \mathcal{D}(g^x, g^y, g^{xy}) = 1] - \Pr[x, y, z \xleftarrow{\mathcal{R}} \mathbb{Z}_p : \mathcal{D}(g^x, g^y, g^z) = 1] \right|$$

Then DDH assumption for G assumes that for any efficient algorithm $\mathcal{D}$, its DDH-advantage is negligible.

Theorem 1 *The scheme described in Section 3.3.1 is IND-CCVA secure assuming that the DDH assumption holds in G.*

Proof : The proof goes by reduction which shows that if an adversary is able to break the IND-CCVA security, it can be used to solve the DDH problem. Let us assume, there is an adversary $\mathcal{A}$ which can break the IND-CCVA security of the scheme. Using $\mathcal{A}$, we can construct an algorithm $\mathcal{B}$ that solves the DDH problem.

$\mathcal{B}$ is given as input a 4-tuple (g, g^a, g^b, Z), where a, b are chosen randomly from $\mathbb{Z}_p$. The task of $\mathcal{B}$ is to determine whether Z is equal to g^{ab} or Z is a random element of G. $\mathcal{B}$ solves this problem by interacting with $\mathcal{A}$ in the IND-CCVA game as follows.

- **Simulation of Key Generation (KG):** $\mathcal{B}$ proceeds as follows:
  - Sets $g_1 = g$.
  - Chooses $s \xleftarrow{\mathcal{R}} \mathbb{Z}_p$ and sets $g_2 = g_1^s$.
  - Chooses $x_1, x_2 \xleftarrow{\mathcal{R}} \mathbb{Z}_p$ and sets $c = g_1^{x_1} g_2^{x_2}$.
  - Sets $h = g^b$.
  - Finally the 4-tuple (g_1, g_2, c, h) is made available as public key to $\mathcal{A}$ by $\mathcal{B}$.

- **Simulation of Ciphertext Verification Oracle for Ciphertext Validity Check:**
  - Knowledge of (x_1, x_2) ensures that $\mathcal{B}$ can perfectly answer the ciphertext verification queries asked by $\mathcal{A}$.

- **Simulation of Challenge Ciphertext:**
  - In Challenge Phase, $\mathcal{A}$ chooses and outputs two messages m_0 and m_1 to $\mathcal{B}$.
  - $\mathcal{B}$ then chooses a bit $\tau \xleftarrow{\mathcal{R}} \{0,1\}$ and it proceeds to encrypt m_τ.
  - $\mathcal{B}$ sets
$$u_1 = g^a,\ u_2 = (g^a)^s,\ e = Z \cdot m_\tau \text{ and } v = (g^a)^{x_1}(g^a)^{sx_2}.$$
  - The challenge ciphertext (u_1, u_2, e, v) is given to $\mathcal{A}$ by $\mathcal{B}$.

Finally in the Guess Phase, $\mathcal{A}$ answers a bit τ'. If $\tau = \tau'$ then $\mathcal{B}$ announces the input instance to be a valid DDH tuple, else ($\tau \neq \tau'$) $\mathcal{B}$ announces invalid tuple. This completes the description of $\mathcal{B}$. We show that

$$\mathcal{A}dv(\mathcal{B}) = \mathcal{A}dv(\mathcal{A}).$$

For this it is enough to show that simulation of challenge ciphertext is perfect given a valid DDH instance. This is true as for valid DDH tuple (i.e., $Z = g^{ab}$) we have

- $u_1 = g^a = g_1^a$.
- $u_2 = (g^a)^s = (g^s)^a = g_2^a$.
- $e = Z \cdot m_\tau = g^{ab} \cdot m_\tau = (g^b)^a \cdot m_\tau = h^a \cdot m_\tau$.
- $v = (g^a)^{x_1}(g^a)^{sx_2} = (g^{x_1}g^{sx_2})^a = c^a$.

Thus the simulation of challenge ciphertext is perfect. This proves the theorem. ∎

Lemma 1 *The scheme described in Section 3.3.1 is not IND-CCA secure.*

Proof : In IND-CCA game, if $\mathcal{C} = (u_1, u_2, e, v)$ be the challenge ciphertext, adversary $\mathcal{A}$ chooses any message $m' \neq 1$ (identity in G) and creates another ciphertext $\mathcal{C}' = (u_1, u_2, m'e, v)$ which is indeed different than challenge ciphertext. Decryption oracle returns $m'm$ if $\mathcal{C}'$ is queried to it. $\mathcal{A}$ then easily calculates the original message by calculating $m'mm'^{-1} = m$. Hence, the lemma. ∎

3.4 The Separating Scheme (Known): IND-CPA secure but not IND-CCVA secure

It is well-known that textbook RSA does not hide partial information about the plaintext, is malleable, and is also insecure against chosen ciphertext attack. Indeed, textbook RSA is never used in practice, precisely because of these well-known weaknesses. Instead, what people actually use is textbook RSA with a few modifications attempt to fix these problems.

One idea that is often advocated to improve the security of textbook RSA is to use a randomized "encoding" or "padding" scheme. That is, we encrypt m as $C = f(m, r)^e$, where

$f(m, r)$ encodes the message m using some random bits r. Note that f is not a cryptographic encoding: it is easy for anyone to compute m from $f(m, r)$. The hope is that this enhancement improves the security of RSA. However, if one is not extremely careful, the resulting scheme may become insecure.

One simple way to define $f(m, r)$ is just to concatenate the two bit strings m and r. This is a popular idea. RSA, Inc. has a very popular encryption function, called PKCS #1, which did essentially this until the well-known attack by Bleichenbacher [14] had surfaced. This encryption function is used by the security protocol SSL over internet.

In literature, Bleichenbacher's attack on SSL has been termed as chosen ciphertext attack on RSA's PKCS #1. But we observe that, his attack is actually a chosen ciphertext verification attack. We first describe briefly the RSA encryption standard PKCS #1; refer to [54] for details. It has three block formats: Block types 0 and 1 are reserved for digital signatures, and block type 2 is used for encryption. As we are interested in encryption only, we describe the block 2.

- **KG(1^λ):** Choose primes p, q ($4k$ bit each) and compute $n = pq$ (n is k byte number). Choose e, d, such that $ed \equiv 1 \pmod{\phi(n)}$. The public key, PK, is (n, e) and the secret key, SK, is (p, q, d).
- **ENC(m, PK):** A data block D, consisting of $|D|$ bytes, is encrypted as follows:
  - First, a padding string PS, consisting of $k - 3 - |D|$ nonzero bytes, is generated pseudo-randomly (the byte length of PS is atleast 8).
  - Now, the encryption block $EB = 00||02||PS||00||D$ is formed, is converted into an integer x, and is encrypted with RSA, giving the ciphertext $c = x^e \pmod{n}$.
- **DEC(c, SK, PK)** A Ciphertext c is decrypted as follows:
  - Compute $x' = c^d \pmod{n}$.
  - Converts x' into an encryption block EB'.
  - Check, if the encryption block is PKCS *conforming* (An encryption block EB consisting of k bytes, $EB = EB_1||\ldots||EB_k$, is called PKCS conforming, if it satisfies the following conditions: $EB_1 = 00$, $EB_2 = 02$, EB_3 through EB_{10} are nonzero and at least one of the bytes EB_{11} through EB_k is 00).
  - If the encryption block is PKCS conforming, then output the data block; otherwise an error sign.

3.4.1 Security

It is well-known that the least significant bit of textbook RSA encrypted message is as secure as the whole message [48, 2]. In particular, there exists an algorithm that can decrypt a ciphertext if there exists another algorithm that can predict the least significant bit of a message given only the corresponding ciphertext and the public key. Håstad and Näslund

extended this result to show that all individual RSA bits are secure [50].

Bleichenbacher's attack assumes that the adversary has access to an oracle that, for every ciphertext, returns whether the corresponding plaintext is PKCS conforming. If the plaintext is not PKCS conforming, the oracle outputs an error sign. Given just these error signs, because of specific properties of PKCS #1, Bleichenbacher showed how a very clever program can decrypt a target ciphertext (the oracle answer will reveal the first two bytes of the corresponding plaintext of the chosen ciphertext). Though, at this point the algorithm of Håstad and Näslund can use this oracle to decrypt the target ciphertext, Bleichenbacher's attack, different from Håstad and Näslund, was aimed at minimizing the number of oracle queries; thus, showing the practicality of the attack.

Hence, all the attacker needs is the verification about the validity of the chosen ciphertext (and not the corresponding whole plaintext). Thus this is clearly a chosen ciphertext verification attack.

3.5 Separating Schemes: Generic Constructions

In this section we provide generic constructions of public key encryption schemes that are

- IND-CPA secure but not IND-CCVA secure,
- IND-CCVA secure but not IND-CCA secure,
- IND-CCA1 secure but not IND-CCVA secure.

The constructions are based on the existence of (enhanced) trapdoor permutations (see Appendix C in [47]). We refer the reader to [47] (pages 413-422) for the encryption schemes, based on the existence of trapdoor permutations, that are IND-CPA secure but not IND-CCA secure with the property that $\cup_{k\in\mathbf{K}}\text{ENC}(\mathbf{M}) = \mathbf{C}$. Constructions, based on enhanced one-way trapdoor permutation, that are IND-CCA1 secure but not IND-CCA secure are also given in [47] (pages 452-461).

3.5.1 Generic Construction: IND-CPA secure but not IND-CCVA secure

Let $\mathcal{E}^{CPA}$ be a public key encryption scheme described by the key generation algorithm KG^{CPA}, encryption algorithm ENC^{CPA} and decryption algorithm DEC^{CPA}. Now define a new public key encryption $\mathcal{E}$ as follows

- KG: Same as KG^{CPA}.
- ENC: Encryption of a message m under a public key PK is give as

$$c = c_1||c_2 = \text{ENC}^{CPA}(m, PK)||\text{ENC}^{CPA}(m, PK)$$

- DEC: Decryption of a ciphertext $c = c_1||c_2$ with the corresponding secret key SK will proceed as follows:
  - $m'_1 \leftarrow \text{DEC}^{CPA}(c_1, SK, PK)$
  - $m'_2 \leftarrow \text{DEC}^{CPA}(c_2, SK, PK)$
  - If $m'_1 = m'_2$, return m'_1, else
  - return $\perp$

Theorem 2 *If $\mathcal{E}^{CPA}$ is IND-CPA secure then $\mathcal{E}$ is also IND-CPA secure.*

Proof : Straightforward.

Lemma 2 *Encryption scheme $\mathcal{E}$ is not IND-CCVA secure.*

Proof : We construct an efficient IND-CCVA adversary $\mathcal{A}$ against $\mathcal{E}$. In the challenge phase of the IND-CCVA security game, $\mathcal{A}$ outputs two equal length messages m_0, m_1 and request the challenger to encrypt one of the message. The challenger picks a challange bit $b \in \{0, 1\}$ at random, encrypts m_b under the public key PK, and returns the challenge ciphertext $c_b = c_{b_1}||c_{b_2}$. The adversary $\mathcal{A}$ now picks one of the message, say m_1, and computes $c_1 = ENC^{CPA}(m_1, PK)$. $\mathcal{A}$ now submits the modified ciphertext $\bar{c} = c_1||c_{b_2}$ to the Chosen Ciphertext Verification Oracle. Now $\mathcal{A}$ will return 1 if and only if the oracle returns 1. It is easy to verify that $\mathcal{A}$'s guess is correct with probability 1. Hence the encryption scheme $\mathcal{E}$ is not IND-CCVA secure.

3.5.2 Generic Construction: IND-CCVA secure but not IND-CCA secure

In [47] (pages 413-422), the one way trapdoor permutation based constructions that are IND-CPA secure but not IND-CCA secure also possesses the following property

$$\cup_{k \in \mathbf{K}} \text{ENC}(\mathbf{M}) = \mathbf{C}.$$

Let us denote this scheme by $\mathcal{E} = (\text{KG}, \text{ENC}, \text{DEC})$. The IND-CCVA adversary against $\mathcal{E}$ will not gain anything new by using the verification oracle and thus $\mathcal{E}$ is IND-CCVA secure but not IND-CCA secure. But we assumed in this article to work on schemes that satisfy $\cup_{k \in \mathbf{K}} \text{ENC}(\mathbf{M}) \neq \mathbf{C}$. We now give such a construction.

Let us build a new public key encryption $\hat{\mathcal{E}} = (\text{KG}^{\hat{\mathcal{E}}}, \text{ENC}^{\hat{\mathcal{E}}}, \text{DEC}^{\hat{\mathcal{E}}})$ based on $\mathcal{E}$ as follows.

- $\text{KG}^{\hat{\mathcal{E}}}$: Same as KG.
- $\text{ENC}^{\hat{\mathcal{E}}}$: Encryption of a message m under a public key PK is give as

$$\hat{c} = 1||c, \text{ where } c = \text{ENC}(m, PK).$$

- $\text{DEC}^{\hat{\mathcal{E}}}$: Decryption of a ciphertext $\hat{c}$ with the corresponding secret key SK will proceed as follows:

$$\text{DEC}^{\hat{\mathcal{E}}}(\hat{c}, SK, PK) = \text{DEC}(c, SK, PK) \text{ if } \hat{c} = 1||c, \text{ otherwise return } \perp.$$

It is easy to check that $\hat{\mathcal{E}}$ is IND-CPA secure but not IND-CCA secure with the added property that every ciphertext need not be valid. Since it is trivial to distinguish valid ciphertexts from invalid ciphertexts (by just looking at the most significant bit), CCVA oracle does not give any extra advantage to the adversary and thus $\hat{\mathcal{E}}$ is IND-CCVA secure.

3.5.3 Generic Construction: IND-CCA1 secure but not IND-CCVA secure

In this section, we give a generic construction of IND-CCA1 secure encryption scheme which is not IND-CCVA secure. Let $\mathcal{E}^{CCA1}$ be a IND-CCA1 secure encryption scheme. Let (PK, SK) be the public key-secret key pair, ENC^{CCA1} be the encryption algorithm and DEC^{CCA1} be the decryption algorithm of $\mathcal{E}^{CCA1}$. We construct an encryption scheme, say $\mathcal{E}$ from $\mathcal{E}^{CCA1}$ whose public key-secret key pair is (PK, SK). Encryption algorithm of $\mathcal{E}$, say ENC, takes a message m and outputs ciphertext c.

- $c = c_1||c_2 \leftarrow \text{ENC}(m, PK)$

where $c_1 \leftarrow \text{ENC}^{CCA1}(m, PK)$, $c_2 \leftarrow \text{ENC}^{CCA1}(m, PK)$, and $PK = PK_1$. Decryption algorithm of $\mathcal{E}$, say DEC, for an input $c = c_1||c_2$ is defined as following:

- $m'_1 \leftarrow \text{DEC}^{CCA1}(c_1, SK, PK)$
- $m'_2 \leftarrow \text{DEC}^{CCA1}(c_2, SK, PK)$
- If $m'_1 = m'_2$, return m'_1, else
- return $\perp$

Theorem 3 *If $\mathcal{E}^{CCA1}$ is IND-CCA1 secure then $\mathcal{E}$ is also IND-CCA1 secure.*

Proof : Straightforward.

Lemma 3 *The Encryption scheme $\mathcal{E}$ is not IND-CCVA secure.*

Proof : Similar to the proof of lemma 2.

Chapter 4

Construction of Identity Based Signcryption Schemes

In this chapter, we show how to construct an Identity Based Signcryption (IBSC) scheme using an Identity Based Encryption (IBE) and an Identity Based Signature (IBS) schemes. We show that the security of the IBSC scheme–indistinguishability as well as unforgeability– is derived from the security of the underlying IBE and IBS schemes. We have proposed two schemes, IBSC-Scheme1 and IBSC-Scheme2. Both schemes achieve the same level of security, however, the difference lies at the computational efficiency. Compared to Sign-then-Encrypt approach, IBSC-Scheme1 is efficient during signcryption phase only whereas IBSC-Scheme2 is efficient in both phases *viz* signcryption and designcryption. We obtain IBSC-Scheme2 by first extending the An-Dodis-Rabin construction to the Identity Based setting. We then further modify the construction to obtain an efficient construction.

4.1 Introduction

In order to simplify the key management, Shamir proposed an Identity Based Cryptosystem [85] in 1984. In this cryptosystem, unambiguous identity of a user(such as email address, social security number etc.) is used as a public key and the secret key corresponding to a user is issued by a third party called the Private Key Generator (PKG). Since 1984, although Shamir proposed the first Identity Based Signature (IBS) scheme in his proposal of Identity Based Cryptosystem, a satisfactory solution for Identity Based Encryption (IBE) eluded cryptographers till the turn of the millennium [23]. The posed demand was unfulfilled until 2000 when Sakai-Ohgishi-Kashara [83] and 2001 when Boneh and Franklin [15] proposed an IBE scheme from bilinear pairing on Elliptic Curves. In 2001 again, Cocks [29] proposed an IBE scheme based on quadratic residuosity problem modulo an RSA composite modulus.

In 1997, Zheng [94] proposed a new primitive *viz* signcryption in which encryption and signature are done simultaneously at a much lower computational cost and communication overhead than the Sign-then-Encrypt approach. The scheme in [94] was not formally proved to be secure since no formal notion of security was proposed then. It was only in PKC 2002 that Baek, Steinfeld and Zheng [6] introduced a formal notion of security for signcryption.

Since the introduction of the primitive, several schemes have been proposed [3, 65, 64, 7, 25, 34, 33, 69, 26]. In 2009, Matsuda-Matsuura-Schuldt [69] gave several simple but efficient constructions of signcryption schemes using existing primitives. In one of the constructions, they introduced the notion of *signcryption composable* and show how, in this case, a signature scheme and an encryption scheme, if they use *shared randomness*, can be combined to achieve higher efficiency than a simple composition. Signcryption composability finally yields efficient insider secure schemes in the standard model. Moreover, they proposed other efficient constructions using tag-based Key Encapsulation (tag-based KEM) mechanism and Data Encapsulation Mechanism (DEM). These constructions also are efficient and insider secure in the standard model. (Also, [69, 35] gives a nice account of some previous work on signcryption and has an extensive bibliography.) In 2011, Chiba et. al. [26] proposed some generic constructions which were insider secure in the standard model. They used Sign-then-Encrypt paradigm which relies upon chosen-ciphertext-secure tag-based KEM, a chosen-ciphertext-secure DEM that has a one-to-one property, and a strongly unforgeable signature scheme.

In this chapter, we consider an efficient construction of an *Identity Based* Signcryption scheme. We will show how to construct an Identity Based Signcryption (IBSC) scheme using any Identity Based Encryption (IBE) scheme and Identity Based Signature (IBS) scheme. Our construction differs from those of [69, 26] in the sense that we do not use the sign-then-encrypt or encrypt-then-sign paradigm. In fact, our first construction IBSC-Scheme1 allows signature and encryption to be done in parallel during signcryption for increased efficiency. Whereas our second construction allows signature and encryption to be done in parallel during signcryption while decryption and verification can be done in parallel during designcryption. IBSC-Scheme2 security of the resulting IBSC scheme is inherited from the security results of the underlying IBE and IBS schemes in the random oracle model. We then compare these schemes with the existing schemes which are known to be efficient and secure and show how the resulting schemes compare with the existing ones.

In the public key setting, An, Dodis and Rabin [3] proposed a generic construction of Signcryption scheme using Commit then Encryption and Signature paradigm ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) . Their construction is efficient in the sense that Encryption and Signature can be done in parallel. In this chapter, we show that that their construction can easily be lifted to the Identity Based setting to yield an Identity Based Signcryption scheme. However, like the An-Dodis-Rabin construction, we only obtain a *generalized* IND-CCA secure IBSC scheme from an IND-ID-CCA secure IBE. In fact, we show that this scheme is not IND-CCA secure. To obtain an IND-IBSC-CCA secure Identity Based Signcryption scheme, we modify the preceding construction. We show that this modification *viz* IBSC-Scheme2 yields an *IND-IBSC-CCA secure* signcryption scheme, provided the underlying IBE is *IND-ID-CCA secure*. Finally, we show that our construction yields an efficient identity based signcryption schemes when compared with existing ones.

4.2 Proposed Scheme : IBSC-Scheme1

Let $S_{IBE} = (\text{Setup}^{IBE}, \text{KeyGen}^{IBE}, \text{Encrypt}, \text{Decrypt})$ and $S_{IBS} = (\text{Setup}^{IBS}, \text{KeyGen}^{IBS}, \text{Sign}, \text{Verify})$ be an Identity Based Encryption scheme and an Identity Based Signature scheme respectively. Let l_1 be the bit-length of any message m from the message space $\mathbf{M}$. We require that the bit-length of a random number r from the random space $\mathbf{R}$ also be l. Moreover, let l_2 be the bit-length of the signature s generated by the algorithm 'Sign' of S_{IBS}.

The construction of an Identity Based Signcryption scheme from Identity Based Encryption and Signature schemes has been described in Table 4.1.

<table>
<tr><th colspan="2">Proposed Scheme: IBSC-Scheme1</th></tr>
<tr><td>

Setup(1^λ)

- Choose three cryptographically secure hash functions
$H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$,
$H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_2}$,
$H_3 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$.
- $(MSK^{IBE}, Params^{IBE}) \leftarrow \text{Setup}^{IBE}(1^\lambda)$
- $(MSK^{IBS}, Params^{IBS}) \leftarrow \text{Setup}^{IBS}(1^\lambda)$
- $MSK \leftarrow (MSK^{IBE}, MSK^{IBS})$ and
- $Params \leftarrow (Params^{IBE}, Params^{IBS}, H_1, H_2, H_3)$
- return $(MSK, Params)$

</td><td>

Signcrypt$(m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$

- Choose r randomly from $\mathbf{R}$.
- Let $c' \leftarrow \text{Encrypt}(r, ID_{Rec}, Params^{IBE})$.
- Compute $h_1 = H_1(r, c', ID_{Sen})$.
- Compute $h_2 = H_2(m, c', h_1, ID_{Rec}, ID_{Sen})$.
- Compute $c = H_3(h_1, ID_{Sen}) \oplus m$.
- Let $SK_{ID_{Sen}} = (SK^{IBE}_{ID_{Sen}}, SK^{IBS}_{ID_{Sen}})$.
$(m, s) \leftarrow \text{Sign}(m, SK^{IBS}_{ID_{Sen}}, ID_{Sen}, Params^{IBS})$.
- Compute $d = h_2 \oplus s$.

The Ciphertext will be $\mathcal{C} \equiv (c, c', d)$.

</td></tr>
<tr><td>

KeyGen(ID, MSK)

- $SK^{IBE}_{ID} \leftarrow \text{KeyGen}^{IBE}(ID, MSK^{IBE})$
- $SK^{IBS}_{ID} \leftarrow \text{KeyGen}^{IBS}(ID, MSK^{IBS})$.
- $SK_{ID} \leftarrow (SK^{IBE}_{ID}, SK^{IBS}_{ID})$.

</td><td>

Designcrypt$(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$

- Let $SK_{ID_{Rec}} = (SK^{IBE}_{ID_{Rec}}, SK^{IBS}_{ID_{Rec}})$.
Let $r' \leftarrow \text{Decrypt}(c', SK^{IBE}_{ID_{Rec}}, Params^{IBE})$.
- If r' is $\perp$, return $\perp$, else goto next step
- Compute $h'_1 = H_1(r', c', ID_{Sen})$.
- Compute $m' = H_3(h'_1, ID_{Sen}) \oplus c$.
- Compute $h'_2 = H_2(m', c', h'_1, ID_{Rec}, ID_{Sen})$.
- Compute $s' = h'_2 \oplus d$.
- Let $x \leftarrow \text{Verify}(m', s', ID_{Sen}, Params^{IBS})$.
- If x is true, return (m', s'), else $\perp$.

</td></tr>
</table>

Table 4.1: Construction of IBSC from IBE and IBS schemes

4.3 Security

4.3.1 Message Confidentiality

We will prove our scheme to be IND-IBSC-CCA secure under the random oracle model if the underlying Identity Based Encryption scheme is IND-ID-CCA secure.

Theorem 4 *In the random oracle model, if there exists an IND-IBSC-CCA adversary $\mathcal{A}$ which is able to distinguish ciphertexts during the game of definition 2.9.1 with an advantage ϵ, then there exists an IND-ID-CCA adversary $\mathcal{B}$ that has advantage $\frac{\epsilon}{2}$ against the underlying Identity Based Encryption scheme S_{IBE}.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBE} algorithm of S_{IBE}. We shall show how to construct an IND-ID-CCA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\frac{\epsilon}{2}$ against S_{IBE}. Suppose $\mathcal{B}$ receives public parameters $Params^{IBE}$ from $\mathcal{CH}$. $\mathcal{B}$ chooses an Identity Based Signature scheme S_{IBS} whose public parameters $Params^{IBS}$ are independently generated from the public parameters of S_{IBE}. We can safely assume that $Params^{IBE} \cap Params^{IBS} = \phi$. $\mathcal{B}$ maintains lists L_1, L_2 and L_3 for queries on hash functions H_1, H_2 and H_3. Besides these, $\mathcal{B}$ maintains two other lists $\mathcal{S}_1$ and $\mathcal{S}_2$ for queries on secret keys of different identities corresponding to S_{IBE} and S_{IBS} .

We now explain how requests from $\mathcal{A}$ are treated by $\mathcal{B}$ who plays the role of a challenger to $\mathcal{A}$.

- H_1 queries : For inputs $r^{(i)}$, $c'^{(i)}$ and $ID^{(i)} \in \{0,1\}^*$ from $\mathcal{A}$, $\mathcal{B}$ searches list L_1 for tuple $(r^{(i)}, c'^{(i)}, ID^{(i)}, h_1^{(i)})$. If such tuple exists, $\mathcal{B}$ returns $h_1^{(i)}$ to $\mathcal{A}$, else $\mathcal{B}$ chooses a string uniformly at random from $\{0,1\}^{l_1}$, say $h_1^{(i)}$. $\mathcal{B}$ then adds the tuple $(r^{(i)}, c'^{(i)}, ID^{(i)}, h_1^{(i)})$ into the list L_1 and returns $h_1^{(i)}$ to $\mathcal{A}$.

- H_2 queries : For input $m^{(i)}$, $c'^{(i)}$, $h^{(i)} \in \{0,1\}^{l_1}$, and $ID_1^{(i)}$, $ID_2^{(i)} \in \{0,1\}^*$ from $\mathcal{A}$, $\mathcal{B}$ searches the list L_2 for tuple $(m^{(i)}, c'^{(i)}, h^{(i)}, ID_1^{(i)}, ID_2^{(i)}, h_2^{(i)})$. If such a tuple exists, B returns $h_2^{(i)}$ to $\mathcal{A}$, else $\mathcal{B}$ chooses a string uniformly at random from $\{0,1\}^{l_2}$, say $h_2^{(i)}$. $\mathcal{B}$ then adds the tuple $(m^{(i)}, c'^{(i)}, h^{(i)}, ID_1^{(i)}, ID_2^{(i)}, h_2^{(i)})$ to the list L_2 and returns $h_2^{(i)}$ to $\mathcal{A}$.

- H_3 queries : For inputs $h^{(i)} \in \{0,1\}^{l_1}$ and $ID^{(i)} \in \{0,1\}^*$ from $\mathcal{A}$, $\mathcal{B}$ searches the list L_3 for a tuple $(h^{(i)}, ID^{(i)}, h_3^{(i)})$. If such a tuple exists, $\mathcal{B}$ returns $h_3^{(i)}$ to $\mathcal{A}$, else $\mathcal{B}$ chooses a string $h_3^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$. $\mathcal{B}$ then adds the tuple $(h^{(i)}, ID^{(i)}, h_3^{(i)})$ to the list L_3 and returns $h_3^{(i)}$ to $\mathcal{A}$.

- *KeyGeneration* queries : For an input $ID^{(i)}$ from $\mathcal{A}$, algorithm $\mathcal{B}$ responds to $\mathcal{A}$ in two steps:

  1. $\mathcal{B}$ sends $ID^{(i)}$ to $\mathcal{CH}$. Let $\mathcal{CH}$ returns the corresponding secret key $SK_{ID^{(i)}}^{IBE}$. $\mathcal{B}$ then adds $(ID^{(i)}, SK_{ID^{(i)}}^{IBE})$ into the list $\mathcal{S}_1$.

2. As the constituent Identity Based Signature scheme, S_{IBS}, is chosen by $\mathcal{B}$, so $\mathcal{B}$ generates the secret key $SK^{IBS}_{ID^{(i)}}$ corresponding to $ID^{(i)}$, then adds $(ID^{(i)}, SK^{IBS}_{ID^{(i)}})$ into the list $\mathcal{S}_2$.

$\mathcal{B}$ finally returns $(SK^{IBE}_{ID^{(i)}}, SK^{IBS}_{ID^{(i)}})$ to $\mathcal{A}$.

- *Signcryption* queries : The response to signcryption query for message $m^{(i)}$ corresponding to the receiver's identity $ID_{Rec^{(i)}}$ and the sender's identity $ID_{Sen^{(i)}}$ is as follows :

  1. $\mathcal{B}$ searches the list $\mathcal{S}_2$ for the secret key corresponding to identity $ID_{Sen^{(i)}}$. If it does not exist, $\mathcal{B}$ generates the secret key corresponding to $ID_{Sen^{(i)}}$ using KeyGen^{IBS} algorithm of S_{IBS}. Let $SK^{IBS}_{ID^{(i)}}$ be the corresponding secret key.
  2. $\mathcal{B}$ then chooses a random number $r^{(i)}$ and runs $c'^{(i)} \leftarrow$ Encrypt$(r^{(i)}, ID_{Rec^{(i)}}, Params^{IBE})$.
  3. $\mathcal{B}$ then searches the list L_1 for tuple $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_1^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.
  4. Then, $\mathcal{B}$ searches the list L_2 for tuple $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
  5. Now, $\mathcal{B}$ searches the list L_3 for a tuple $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses $h_3^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$ to the list L_3. $\mathcal{B}$ then computes $c^{(i)} = h_3^{(i)} \oplus m^{(i)}$.
  6. $\mathcal{B}$ then computes $s^{(i)} \leftarrow \text{Sign}(m^{(i)}, SK^{IBS}_{ID_{Sen^{(i)}}}, ID_{Sen^{(i)}}, Params^{IBS})$.
  7. Now, $\mathcal{B}$ computes $d^{(i)} = h_2^{(i)} \oplus s^{(i)}$.

  $\mathcal{B}$ finally sends $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, d^{(i)})$ to $\mathcal{A}$.

  Note that, to signcrypt the message corresponding to the receiver's identity ID_{Rec}, the secret key of ID_{Rec} is not required. Hence a valid ciphertext can be generated without any secret key query for the receiver's identity to $\mathcal{CH}$.

- *Designcryption* queries : For input $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, d^{(i)})$ and $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ (receiver's and sender's identities are $ID_{Rec^{(i)}}$ and $ID_{Sen^{(i)}}$ respectively), $\mathcal{B}$ responds as follows:

  1. $\mathcal{B}$ sends $(c'^{(i)}, ID_{Rec^{(i)}})$ to $\mathcal{CH}$ to decrypt. Let $r^{(i)}$ be the output from the Decrypt algorithm of S_{IBE}.
  2. Then $\mathcal{B}$ searches the list L_1 for tuple $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_1^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.

3. Now, $\mathcal{B}$ searches the list L_3 for a tuple $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses $h_3^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$ to the list L_3.
4. $\mathcal{B}$ then computes $m^{(i)} = h_3^{(i)} \oplus c^{(i)}$.
5. Then, $\mathcal{B}$ searches the list L_2 for tuple $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
6. $\mathcal{B}$ then calculates $s^{(i)} = d^{(i)} \oplus h_2^{(i)}$ and runs Verify$(m^{(i)}, s^{(i)}, ID_{Sen^{(i)}}, Params^{IBS})$.

If the output of Verify algorithm is true then $\mathcal{B}$ sends $(m^{(i)}, s^{(i)})$, else $\perp$, to $\mathcal{A}$.

Once $\mathcal{A}$ decides to enter the challenge phase, it chooses two messages m_0, m_1 of same length and two identities ID_R and ID_S corresponding to the receiver's and sender's identity respectively and sends them to $\mathcal{B}$ which is responded as follows:

1. $\mathcal{B}$ chooses two random strings $r_0, r_1 \in \{0,1\}^{l_1}$ and the receiver's identity ID_R and sends them to $\mathcal{CH}$.
2. $\mathcal{CH}$ then chooses a bit, say b, uniformly at random from $\{0,1\}$ and computes the ciphertext $c'_b \leftarrow$ Encrypt$(r_b, ID_R, Params^{IBE})$ associated to the receiver's identity ID_R.
3. $\mathcal{CH}$ then sends the ciphertext c'_b to $\mathcal{B}$.
4. $\mathcal{B}$ then chooses a bit $v \in \{0,1\}$ uniformly at random. $\mathcal{B}$ then searches the list L_1 for the tuple (r_v, c'_b, ID_S, h_1). If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random h_1 from $\{0,1\}^{l_1}$ and adds (r_v, c'_b, ID_S, h_1) to the list L_1.
5. Now, $\mathcal{B}$ searches the list L_2 for tuple $(m_v, c'_b, h_1, ID_R, ID_S, h_2)$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say h_2, uniformly at random from $\{0,1\}^{l_2}$ and adds $(m_v, c'_b, h_1, ID_R, ID_S, h_2)$ to the list L_2.
6. Then $\mathcal{B}$ searches the list L_3 for the tuple (h_1, ID_S, h_3). If such a tuple doesn't exist, $\mathcal{B}$ chooses a string, say h_3 uniformly at random from $\{0,1\}^{l_1}$ and adds (h_1, ID_S, h_3) to the list L_3.
7. Then $\mathcal{B}$ searches the list $\mathcal{S}_2$ for the secret key corresponding to the identity ID_S. If the secret key does not exist, $\mathcal{B}$ then runs KeyGen algorithm of S_{IBS} . Let the secret key be $SK_{ID_S}^{IBS}$.
8. After that $\mathcal{B}$ runs the Sign algorithm of S_{IBS} on message m_v and identity ID_S to get a signature $s' \leftarrow$ Sign$(m, SK_{ID_{Sen}}^{IBS}, ID_{Sen}, Params^{IBS})$.
9. Now, $\mathcal{B}$ computes $c_v = h_3 \oplus m_v$ and $d = h_2 \oplus s'$ and sends the ciphertext $\mathcal{C}_{b,v} = (c_v, c'_b, d)$ to $\mathcal{A}$.

$\mathcal{A}$ then performs a second series of queries. Queries over *KeyGeneration* (except secret key query on identity ID_R), and *Signcryption* are treated in the same way as it was done before challenge phase. H_1, H_2, H_3 and *Designcryption* queries are treated as follows:

- H_1 queries : For query of the form $(r^{(j)}, c'^{(j)}, ID_{Sen^{(j)}})$, if $(r^{(j)}, c'^{(j)}) = (r_{b'}, c'_b)$ where $b' \in \{0,1\}$, $\mathcal{B}$ outputs b' and sends it to $\mathcal{CH}$ and then aborts the game. Else, it is responded in the same way as it was done before the challenge phase.

- H_2 queries : For query of the form $(m^{(j)}, c'^{(j)}, h_1^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}})$ where $(c'^{(j)}, h_1^{(j)}, ID_{Rec^{(j)}}) = (c'_b, h_1, ID_R)$, $\mathcal{B}$ outputs v and sends it to $\mathcal{CH}$ and then aborts the game. Else, it is responded in the same way as it was done before the challenge phase.

- H_3 queries : For query of the form $(h_1^{(j)}, ID_{Sen^{(j)}})$, if $h_1^{(j)} = h_1$, $\mathcal{B}$ outputs v and sends it to $\mathcal{CH}$ and then aborts the game. Else, it is responded in the same way as it was done before the challenge phase.

- *Designcryption* queries : For queries of the form $\mathcal{C}^{(j)} = (c^{(j)}, c'^{(j)}, d^{(j)})$, if $(c'^{(j)}, ID_{Rec^{(j)}}) = (c', ID_R)$, $\mathcal{B}$ sends $\perp$ to the adversary $\mathcal{A}$. Else, it is responded in the same way as it was done before the challenge phase.

Note that in the above game, $\mathcal{B}$ interacts with $\mathcal{CH}$ as in the real game. Moreover, the game between $\mathcal{A}$ and $\mathcal{B}$ also is simulated correctly.

At the end of the simulation, $\mathcal{B}$ will use the bit guessed by $\mathcal{A}$ to guess the challenge bit with $\mathcal{CH}$. If $\mathcal{A}$ guesses $b' \in \{0,1\}$, $\mathcal{B}$ will also output the same bit *viz* b' to $\mathcal{CH}$. At challenge phase,

- If $b = v$ (the bit chosen by $\mathcal{B}$), the simulation is perfect and the ciphertext, $\mathcal{C}$, produced by $\mathcal{B}$ will be a valid ciphertext of message m_v corresponding to ID_R (receiver's identity) and ID_S (sender's identity). Consequently,

$$\Pr[\mathcal{B} \text{ wins} \mid b = v] = \Pr[\mathcal{A} \text{ wins} \mid \mathcal{C} \text{ is a valid ciphertext}]$$

- If $b \neq v$, then r_v can be thought of as chosen uniformly at random from $\{0,1\}^{l_1} \setminus \{r_b\}$. To $\mathcal{A}$'s view r_v and c' are independent. Hence, the output from H_1 and hence from H_2 and H_3 are, to $\mathcal{A}$'s view, strings uniformly chosen at random from $\{0,1\}^{l_2}$ and $\{0,1\}^{l_1}$ respectively. Thus XOR of h_2 and h_3 with s' and message m_v, i.e. $d = h_2 \oplus s'$ and $c = h_3 \oplus m_v$, results in strings with uniformly distribution in $\{0,1\}^{l_2}$ and $\{0,1\}^{l_1}$ respectively. Thus to $\mathcal{A}$'s view, $C = (c, c', d)$ is a random ciphertext chosen uniformly from the ciphertext space. Hence, in this case

$$\Pr[\mathcal{B} \text{ wins} \mid b \neq v] = \Pr[\mathcal{A} \text{ wins} \mid \mathcal{C} \text{ is a random ciphertext}] = 1/2$$

Therefore,

$$\begin{aligned} \Pr[\mathcal{B} \text{ wins}] &= \Pr[\mathcal{B} \text{ wins } | b = v] \times \Pr[b = v] + \Pr[\mathcal{B} \text{ wins} | b \neq v] \times \Pr[b \neq v] \\ &= \tfrac{1}{2}(\Pr[\mathcal{B} \text{ wins } | b = v]) + \tfrac{1}{2}(\tfrac{1}{2}) \\ &= \tfrac{1}{2}(\Pr[\mathcal{B} \text{ wins } | b = v] + \tfrac{1}{2}) \end{aligned}$$

Hence, advantage of $\mathcal{B} = \mathcal{A}dv(\mathcal{B}) = |\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}|$
$= \frac{1}{2}(|(\Pr[\mathcal{B} \text{ wins } | b = v) - \frac{1}{2}]|) = \frac{1}{2}(|\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}|) = \frac{1}{2}(\mathcal{A}dv(\mathcal{A})) = \frac{\epsilon}{2}$

4.3.2 Ciphertext Unforgeability

We can similarly prove signature unforgeability. We show that our scheme is ESUF-IBSC-CMA secure under the random oracle model provided the underlying IBS scheme is strongly unforgeable under adaptive chosen message attack.

Theorem 5 *In the random oracle model, if there exists an ESUF-IBSC-CMA forger $\mathcal{A}$ which is able to produce a forged ciphertext during the game of definition 2.9.1 with an advantage ϵ, then there exists an SUF-ID-CMA adversary $\mathcal{B}$ which can forge the IBS scheme S_{IBS} with advantage ϵ.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBS} algorithm of S_{IBS}. We shall show how to construct an SUF-ID-CMA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\frac{\epsilon}{2}$ against S_{IBS}. Suppose $\mathcal{B}$ receives public parameters $Params^{IBS}$ from $\mathcal{CH}$. $\mathcal{B}$ chooses an Identity Based Encryption scheme S_{IBE} whose public parameters $Params^{IBE}$ are independently generated from the public parameters of S_{IBS}. We can safely assume that $Params^{IBE} \cap Params^{IBS} = \phi$. $\mathcal{B}$ maintains lists L_1, L_2 and L_3 for queries on hash functions H_1, H_2 and H_3. Besides these, $\mathcal{B}$ maintains two other lists $\mathcal{S}_1$ and $\mathcal{S}_2$ for queries on secret keys of different identities corresponding to S_{IBE} and S_{IBS} .

We now explain how requests from $\mathcal{A}$ are treated by $\mathcal{B}$. The response to H_1, H_2 and H_3 queries are exactly as in the proof of Theorem 4.

- *KeyGeneration* queries : For an input $ID^{(i)}$ from $\mathcal{A}$, algorithm $\mathcal{B}$ responds to $\mathcal{A}$ in two steps:

  1. $\mathcal{B}$ sends $ID^{(i)}$ to $\mathcal{CH}$. Let $\mathcal{CH}$ returns the corresponding secret key $SK^{IBS}_{ID^{(i)}}$. $\mathcal{B}$ then adds $(ID_i, SK^{IBS}_{ID^{(i)}})$ into the list $\mathcal{S}_2$.
  2. As the constituent Identity Based Encryption scheme, S_{IBE}, is chosen by $\mathcal{B}$, so $\mathcal{B}$ generates the secret key $SK^{IBE}_{ID^{(i)}}$ corresponding to $ID^{(i)}$. $\mathcal{B}$ then adds $(ID^{(i)}, SK^{IBE}_{ID^{(i)}})$ into the list $\mathcal{S}_1$.

  $\mathcal{B}$ finally returns $(SK^{IBE}_{ID^{(i)}}, SK^{IBS}_{ID^{(i)}})$ to $\mathcal{A}$.

- *Signcryption* queries : The response to signcryption query for message $m^{(i)}$ corresponding to the receiver's identity $ID_{Rec^{(i)}}$ and the sender's identity $ID_{Sen^{(i)}}$ is as follows :

  1. $\mathcal{B}$ chooses a random number $r^{(i)}$ and runs $c'^{(i)} \leftarrow \text{Encrypt}(r^{(i)}, ID_{Rec^{(i)}}, Params^{IBE})$.
  2. $\mathcal{B}$ then searches the list L_1 for the tuple $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_1^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.
  3. Then, $\mathcal{B}$ searches the list L_2 for the tuple $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.

4. Now, $\mathcal{B}$ searches the list L_3 for a tuple $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses $h_3^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$ to the list L_3. $\mathcal{B}$ then computes $c^{(i)} = h_3^{(i)} \oplus m^{(i)}$.
5. $\mathcal{B}$ sends $(m^{(i)}, ID_{Sen^{(i)}})$ to the challenger $\mathcal{CH}$. Let $s^{(i)}$ be the output from the Sign algorithm of S_{IBS}. $\mathcal{CH}$ then sends $s^{(i)}$ to $\mathcal{B}$.
6. Now, $\mathcal{B}$ computes $d^{(i)} = h_2^{(i)} \oplus s^{(i)}$.

$\mathcal{B}$ finally sends $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, d^{(i)})$ to $\mathcal{A}$.

- *Designcryption* queries : For input $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, d^{(i)})$ and $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ (receiver's and sender's identities are $ID_{Rec^{(i)}}$ and $ID_{Sen^{(i)}}$ respectively), $\mathcal{B}$ responds as follows:

  1. $\mathcal{B}$ searches the list $\mathcal{S}_1$ for the secret key corresponding to identity $ID_{Rec^{(i)}}$. If it does not exist, $\mathcal{B}$ generates the secret key corresponding to $ID_{Rec^{(i)}}$ using KeyGen^{IBE} algorithm of S_{IBE}. Let $SK^{IBE}_{ID_{Rec^{(i)}}}$ be the corresponding secret key.
  2. $\mathcal{B}$ then runs $r^{(i)} \leftarrow \text{Decrypt}(c'^{(i)}, SK^{IBE}_{ID_{Rec^{(i)}}}, Params^{IBE})$.
  3. Then $\mathcal{B}$ searches the list L_1 for tuple $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_1^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.
  4. Now, $\mathcal{B}$ searches the list L_3 for a tuple $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses $h_3^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(h_1^{(i)}, ID_{Sen^{(i)}}, h_3^{(i)})$ to the list L_3.
  5. $\mathcal{B}$ then computes $m^{(i)} = h_3^{(i)} \oplus c^{(i)}$.
  6. Then, $\mathcal{B}$ searches the list L_2 for tuple $(m^{(i)}, c'^{(i)}, h_1^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(m^{(i)}, c'^{(i)}, h_3^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
  7. $\mathcal{B}$ then calculates $s^{(i)} = d^{(i)} \oplus h_2^{(i)}$ and runs Verify$(m^{(i)}, s^{(i)}, ID_{Sen^{(i)}}, Params^{IBS})$.

  If the output of Verify algorithm is true then $\mathcal{B}$ sends $(m^{(i)}, s^{(i)})$, else $\perp$, to $\mathcal{A}$.

  Note that, to designcrypt the ciphertext corresponding to the sender's identity ID_{Sen}, the secret key of ID_{Sen} is not required. Hence a ciphertext can be designcrypted without any secret key query for the receiver's identity to $\mathcal{CH}$.

Once this game is over, $\mathcal{A}$ submits a ciphertext $\mathcal{C} = (c, c', d)$ corresponding to the receiver's identity ID_R and the sender's identity ID_S such that the secret key corresponding to ID_S have not been queried earlier. $\mathcal{B}$ then responds as follows:

1. $\mathcal{B}$ decrypts c' corresponding to the receiver's identity ID_R. Let r be the output of the decryption algorithm.

2. $\mathcal{B}$ then searches the list L_1 for a tuple of the form (r, c', ID_S, h_1). If no such tuple exists, $\mathcal{B}$ chooses a uniformly at random a string, say h_1, from $\{0,1\}^{l_1}$ and adds (r, c', ID_S, h_1) to the list L_1.

3. Then $\mathcal{B}$ searches the list L_3 for tuple (h_1, ID_S, h_3). If such a tuple does not exist, $\mathcal{B}$ chooses h_3, say, uniformly at random from $\{0,1\}^{l_1}$ and adds (h_1, ID_S, h_3) to the list L_3 and then computes $m = c \oplus h_3$.

4. $\mathcal{B}$ then searches the list L_2 for a tuple $(m, c', h_1, ID_R, ID_S, h_2)$. If such a tuple does not exist, $\mathcal{B}$ chooses h_2 uniformly at random from $\{0,1\}^{l_2}$ and adds $(m, c', h_3, ID_R, ID_S, h_2)$ to the list L_2 and computes $s = d \oplus h_2$.

Finally, $\mathcal{B}$ submits (m, s) corresponding to the sender's identity ID_S to the challenger $\mathcal{CH}$.

It is clear if $\mathcal{C} = (c, c', d)$ is a valid ciphertext corresponding to receiver's identity ID_R and sender's identity ID_S, then (m, s) is a valid message-signature pair for the sender's identity ID_{Sen}. Note that, the validity of ciphertext $\mathcal{C}$ in the game of ESUF-IBSC-CMA ensures that $\mathcal{A}$ has not queried on (m, ID_S) such that s is a valid signature on message m and hence $\mathcal{B}$ also has not queried on (m, ID_S) to the challenger $\mathcal{CH}$ whose output is s. Therefore, whenever $\mathcal{A}$ produces a forged ciphertext, $\mathcal{B}$ also produces a forged message-signature pair in the SUF-ID-CMA game against S_{IBS}.

Hence, advantage of $\mathcal{B} = \mathcal{A}dv(\mathcal{B}) = \Pr[\mathcal{B} \text{ wins}] = \epsilon$.

4.4 Efficiency

1. Time Efficiency : Our proposed scheme is based upon "Encrypt and Sign" paradigm, where a random number is encrypted instead of a message. Hence, Encrypt and Sign can be run in parallel in the Signcrypt algorithm. Let t_E, t_S, t_D, t_V and t_H be the time taken by the Encrypt, Sign, Decrypt, Verify and Hash algorithms respectively. Then, assuming that the Sign and Encrypt are run concurrently, the time taken by our scheme in Signcrypt will be $T_{SC} = \max(t_E, t_S) + 3t_H$; whereas in the Sign-then-Encrypt approach, the total time taken will be $t_E + t_S$. Moreover, the time taken by our scheme in Designcrypt will be $T_{DSC} = t_D + t_V + 3t_H$. In general, $t_H << (t_E \text{ or } t_S)$.

2. Space Efficiency : In many cases, in practice, the ciphertext length bears (approx.) a constant ratio with the plaintext. This is also the case with many signature schemes. Let the output length of the Encrypt algorithm be (at most) αl_1, where l_1 is the bit length of message m. Let the output length of the signature corresponding to an l_1 bit message be (at most) $l_2 = \beta l_1$. Hence, the total length of ciphertext will be (at most) $(\alpha + \beta + 1)l_1$. But in the Sign-then-Encrypt approach, ciphertext length will be, roughly, $\alpha(\beta + 1)l_1$. Hence, our scheme is likely to produce a shorter ciphertext length compared to the Sign-then-Encrypt approach if $\alpha \geq \frac{1}{\beta} + 1$.

4.5 Comparisons

Using our generic method, we composed two IBSC scheme - first one by composing Boneh-Franklin Identity Based Encryption (BF-IBE) [15] with Shamir's Identity Based Signature (SH-IBS) [85] scheme and the second one by composing Boneh-Franklin IBE [15] with Kurosawa-Heng Identity Based Signature (KH-IBS) ([59], page 113 of [63]) scheme. We compared these schemes with the Identity Based Signcryption (IBSC) schemes proposed by Boyen [18], Chen-Malone-Lee [25] and Barreto et. al. [7]. BF-IBE + SH-IBS (Boneh-Franklin IBE and Shamir IBS) has more than double ciphertext overhead and BF-IBE + KH-IBS (Boneh-Franklin IBE and Kurosawa-Heng IBS) has almost double ciphertext overhead than IBSC schemes proposed by Boyen, Chen-Malone-Lee and Barreto. In case of time efficiency, both schemes (BF-IBE + SH-IBS and BF-IBE + KH-IBS) take less time in Signcrypt (note to remember that in our method, in Signcrypt, Encrypt and Sign algorithm can be run in parallel) and Designcrypt compared to Boyen and Chen-Malone-Lee IBSC scheme. Barreto's scheme has lower cost of computation in Signcrypt than that of BF-IBE + SH-IBS and BF-IBE + KH-IBS but in Designcrypt, BF-IBE + SH-IBS has lower and BF-IBE + KH-IBS has almost equal cost of computation than that of Barreto. Summary of the efficiency comparisons has been given in table 4.2.

Table 4.2: Efficiency Comparisons

Scheme	Signcrypt				Designcrypt			Ciphertext Overhead
	E	P	SM		E	P	SM	
Boyen IBSC [18]	1	1	3		0	4	2	$2\|G_1\| + \|ID\| + \|M\|$
Chen-Malone-Lee IBSC [25]	0	1	3		0	3	1	$2\|G_1\| + \|ID\| + \|M\|$
Barreto et. al. IBSC [7]	1	0	3		1	2	1	$2\|G_1\| + \|M\|$
Different IBSC constructed using our generic method								
BF-IBE [15] + SH-IBS [85]	1	1	1		2	1	1	$\|G_1\| + 2\|\mathbb{Z}_N^*\| + 3\|M\|$
BF-IBE [15] + KH-IBS ([59], [63])	1	1	2		1	2	1	$2\|G_1\| + \|\mathbb{Z}_q^*\| + 3\|M\|$

- E denotes number of exponentiation.
- P denotes number of pairing. We assume $e : G_1 \times G_1 \rightarrow G_T$, if e is a symmetric bilinear map, else $e : G_1 \times G_2 \rightarrow G_T$, in case of asymmetric bilinear map.
- SM denotes number of scalar multiplication of a point on elliptic curve.
- $|G_1|$ denotes the bit length of an element in group G_1 used in pairing.
- $|G_2|$ denotes the bit length of an element in group G_2 used in pairing.
- $|G_T|$ denotes the bit length of an element in group G_T used in pairing.
- $|ID|$ denotes the bit length of an identity.
- $|M|$ denotes the message length.
- $|Z_N^*|$ denotes the length of an element in Z_N^* where $N = pq$ is the product of two prime numbers p and q.
- $|Z_q^*|$ denotes the length of an element of Z_q^*. Here q is a prime number.

4.6 Extension of An-Dodis-Rabin Construction

In [3], An-Dodis-Rabin gave a generic construction of signcryption using *Commitment*, *Encryption* and *Signature* algorithms. Their scheme is in the traditional PKC setting. It can easily been seen that their construction can be lifted to construct an Identity Based Signcryption. Using Commitment, Identity Based Encryption and Identity Based Signature in their construction, we show that similar to the construction in [3] one can obtain an Identity Based Signcryption scheme.

4.6.1 ID-Based An-Dodis-Rabin Construction

Let $S_{COMM} = (\text{Setup}^{COMM}, \text{Commit}, \text{Open})$ be a non-interactive Commitment scheme, $S_{IBE} = (\text{Setup}^{IBE}, \text{KeyGen}^{IBE}, \text{Encrypt}, \text{Decrypt})$ be an Identity Based Encryption scheme, $S_{IBS} = (\text{Setup}^{IBS}, \text{KeyGen}^{IBS}, \text{Sign}, \text{Verify})$ be an Identity Based Signature scheme and m be the message.

<table>
<tr><th colspan="2">Construction Using Commit then Encrypt and Sign Paradigm</th></tr>
<tr><td>Setup(1^λ)
$CK \leftarrow \text{Setup}^{COMM}(1^\lambda)$
$(Params^{IBE}, MSK^{IBE}) \leftarrow \text{Setup}^{IBE}(1^\lambda)$
$(Params^{IBS}, MSK^{IBS}) \leftarrow \text{Setup}^{IBS}(1^\lambda)$
$Params = (CK, Params^{IBE}, Params^{IBS})$
$MSK = (MSK^{IBE}, MSK^{IBS})$
return $(Params, MSK)$</td><td>Signcrypt($m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params$)
$(c, d) \leftarrow \text{Commit}(m, CK)$
$e \leftarrow \text{Encrypt}(d, ID_{Rec}, Params^{IBE})$
$\sigma \leftarrow \text{Sign}(c, SK^{IBS}_{Sen}, ID_{Sen}, Params^{IBS})$
$\mathcal{C} = (e, (c, \sigma))$
return $\mathcal{C}$</td></tr>
<tr><td>KeyGen(ID, MSK)
$SK^{IBE}_{ID} \leftarrow \text{KeyGen}^{IBE}(ID, MSK^{IBE})$
$SK^{IBS}_{ID} \leftarrow \text{KeyGen}^{IBS}(ID, MSK^{IBS})$
$SK_{ID} = (SK^{IBE}_{ID}, SK^{IBS}_{ID})$.
return SK_{ID}</td><td>Designcrypt($\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params$)
Let $SK_{ID_{Rec}} = (SK^{IBE}_{ID_{Rec}}, SK^{IBS}_{ID_{Rec}})$.
$d' \leftarrow \text{Decrypt}(e, SK^{IBE}_{ID_{Rec}}, ID_{Rec}, Params^{IBE})$
$x \leftarrow \text{Verify}(c, \sigma, ID_{Sen}, Params_{IBS})$
If x is false or d' is $\perp$, return $\perp$ else
$m' \leftarrow \text{Open}((c, d'), CK)$
if m' is $\perp$, return $\perp$, else
return $(m', (c, d', \sigma))$.</td></tr>
</table>

Table 4.3: Construction of IBSC from IBE and IBS schemes using $\mathcal{C}t\mathcal{E}\&\mathcal{S}$ Paradigm

As in [3], one can prove the following. The proofs are similar and are omitted.

Theorem 6 *If Commitment has the hiding property and the Identity Based Encryption scheme is IND-ID-gCCA secure, then the Identity Based Signcryption scheme obtained by the extended An-Dodis-Rabin method is IND-IBSC-gCCA secure.*

Remark: See chapter 2, sections 2.7 and 2.9 for the definitions of IND-ID-gCCA secure IBE scheme and IND-IBSC-gCCA secure IBSC schemes.

Theorem 7 *If Commitment has the relaxed binding property and the Identity Based Signature scheme is SUF-ID-CMA secure, then the Identity Based Signcryption scheme constructed above is also ESUF-IBSC-CMA secure.*

We will now show that the IBSC scheme obtained above using the An-Dodis-Rabin construction is *not* IND-IBSC-CCA secure in the insider security model.(This was also observed for the original scheme in [3])

- **Attack on Confidentiality in the Insider Security Model**
  Consider the IND-CCA game between the challenger and the adversary $\mathcal{A}$. Let $(\mathcal{C} = (e, (c, \sigma)))$ be the challenge ciphertext obtained during the IND-IBSC-CCA game corresponding to the receiver's identity ID_{Rec} and sender's identity ID_{Sen}. In the insider security model, we may assume that the adversary $\mathcal{A}$ knows the sender's secret key $SK^{IBS}_{ID_{Sen}}$. Hence, assuming the signature algorithm be probabilistic, $\mathcal{A}$ can obtain a different signature on c, say σ'. Now, designcryption of $(\mathcal{C}' = (e, (c, \sigma')))$ corresponding to the receiver's identity ID_{Rec} and the sender's identity ID_{Sen} will yield the same message, say m, that is obtained from the designcryption of $(\mathcal{C} = (e, (c, \sigma)))$ corresponding to ID_{Rec} and ID_{Sen}. By querying the designcryption oracle, $\mathcal{A}$ easily wins the game.

4.7 A Modified Scheme : IBSC-Scheme2

As observed above, the construction in subsection 4.6.1 yields only an IND-IBSC-gCCA secure scheme from an IND-ID-CCA secure IBE scheme. Thus to obtain an IND-IBSC-CCA secure IBSC scheme we need to modify the construction. This is done below. The modified scheme is *no longer* an instantiation of the extended An-Dodis-Rabin construction.

Let S_{IBE} = (Setup^{IBE},KeyGen^{IBE},Encrypt,Decrypt) and S_{IBS} = (Setup^{IBS},KeyGen^{IBS},Sign, Verify) be an Identity Based Encryption scheme and Identity Based Signature scheme respectively. Let l_1 be the bit-length of any message m from the message space $\mathbf{M}$. We require that the bit-length of a random number r from the random space $\mathbf{R}$ also be l. Moreover, let l_2 be the bit-length of the signature s generated by the algorithm 'Sign' of S_{IBS}.

The construction of an IBSC scheme from IBE and IBS schemes is described in Table 4.4.

Proposed Scheme: IBSC-Scheme2	
Setup(1^λ) • Choose two cryptographically secure hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$, $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_1}$ • $(Params^{IBE}, MSK^{IBE}) \leftarrow \text{Setup}^{IBE}(1^\lambda)$ • $(Params^{IBS}, MSK^{IBS}) \leftarrow \text{Setup}^{IBS}(1^\lambda)$ • $Params = (Params^{IBE}, Params^{IBS}, H_1, H_2)$ • $MSK = (MSK^{IBE}, MSK^{IBS})$ • return $(Params, MSK)$	**Signcrypt**$(m, SK_{ID_{Sen}}, ID_{Rec}, ID_{Sen}, Params)$ • Choose $r \in \mathbf{R}$ • $h_1 = H_1(m, r, ID_{Rec}, ID_{Sen})$ • $c' = \text{Encrypt}(r, ID_{Rec}, Params^{IBE})$ • $\sigma = \text{Sign}(h_1, SK_{ID_{Sen}}, ID_{Sen}, Params^{IBS})$ • $h_2 = H_2(r, c', h_1, \sigma, ID_{Rec}, ID_{Sen})$ • $c = h_2 \oplus m$ • $\mathcal{C} \equiv (c, c', h_1, \sigma)$
KeyGen(ID, MSK) • $SK_{ID}^{IBE} \leftarrow \text{KeyGen}^{IBE}(ID, MSK^{IBE})$ • $SK_{ID}^{IBS} \leftarrow \text{KeyGen}^{IBS}(ID, MSK^{IBS})$. • $SK_{ID} = (SK_{ID}^{IBE}, SK_{ID}^{IBS})$. • return SK_{ID}	**Designcrypt**$(\mathcal{C}, SK_{ID_{Rec}}, ID_{Rec}, ID_{Sen}, Params)$ • Let $SK_{ID_{Rec}} = (SK_{ID_{Rec}}^{IBE}, SK_{ID_{Rec}}^{IBS})$. • $r' = \text{Decrypt}(c', SK_{ID_{Rec}}, ID_{Rec}, Params^{IBE})$ • $x \leftarrow \text{Verify}(h_1, \sigma, ID_{Sen}, Params^{IBS})$ • If r' is $\perp$ or x is false, return $\perp$, else • compute $h_2' = H_2(r', c', h_1, \sigma, ID_{Rec}, ID_{Sen})$ • compute $m' = h_2' \oplus c$ • check $h_1 \stackrel{?}{=} H_1(m', r', ID_{Rec}, ID_{Sen})$ • If the above step is not correctly verified, return $\perp$, else • return $(m', s) = (m', (r', h_1, \sigma))$.

Table 4.4: Construction of IBSC from IBE and IBS schemes.

Remark: Note that, Encrypt and Sign can be run in parallel in Signcrypt. Moreover, Decrypt and Verify also can be run in parallel in Designcrypt. As a result, this construction is more efficient than the previous construction proposed in section 4.2.

4.8 Security of the Modified Scheme

4.8.1 Message Confidentiality

Theorem 8 *Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary which can break our scheme in the IND-IBSC-CCA game with an advantage ϵ in the random oracle model. Then there exists a PPT adversary $\mathcal{B}$ which can break S_{IBE} (Identity Based Encryption scheme used) in the IND-ID-CCA game with an advantage at least $\frac{\epsilon}{2}$.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBE} algorithm of S_{IBE}.

We shall show how to construct an IND-ID-CCA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\frac{\epsilon}{2}$ against S_{IBE}. Suppose $\mathcal{B}$ receives public parameters $Params^{IBE}$ from $\mathcal{CH}$. $\mathcal{B}$ chooses an Identity Based Signature scheme S_{IBS} whose public parameters $Params^{IBS}$ are independently generated from the public parameters of S_{IBE}. We can safely assume that $Params^{IBE} \cap Params^{IBS} = \phi$. $\mathcal{B}$ maintains lists L_1 and L_2 for queries on hash functions H_1, H_2. Besides these, $\mathcal{B}$ maintains two other lists $\mathcal{S}_1$ and $\mathcal{S}_2$ for queries on secret keys of different identities corresponding to S_{IBE} and S_{IBS} .

We now explain how requests from $\mathcal{A}$ are treated by $\mathcal{B}$ who plays the role of a challenger to $\mathcal{A}$.

- H_1 queries : For inputs $m^{(i)}$, $r^{(i)}$, $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ from $\mathcal{A}$, $\mathcal{B}$ searches the list L_1 for the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple exists, $\mathcal{B}$ returns $h_1^{(i)}$ to $\mathcal{A}$, else $\mathcal{B}$ randomly selects $h_1^{(i)}$ from $\{0,1\}^{l_1}$ and adds the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1 and returns $h_1^{(i)}$ to $\mathcal{A}$.

- H_2 queries : For inputs $r^{(i)}$, $c'^{(i)}$, $h_1^{(i)}$, $\sigma^{(i)}$, $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ from $\mathcal{A}$, $\mathcal{B}$ searches the tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ in the list L_2. If such a tuple exists, $\mathcal{B}$ returns $h_2^{(i)}$ to $\mathcal{A}$, else $\mathcal{B}$ randomly selects $h_2^{(i)}$ from $\{0,1\}^{l_1}$ and adds the tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2. $\mathcal{B}$ then returns $h_2^{(i)}$ to $\mathcal{A}$.

- *KeyGeneration* queries : For an input $ID^{(i)}$ from $\mathcal{A}$, algorithm $\mathcal{B}$ responds to $\mathcal{A}$ in two steps:

  1. $\mathcal{B}$ sends $ID^{(i)}$ to $\mathcal{CH}$. Let $\mathcal{CH}$ returns the corresponding secret key $SK_{ID^{(i)}}^{IBE}$. $\mathcal{B}$ then adds $(ID^{(i)}, SK_{ID^{(i)}}^{IBE})$ into the list $\mathcal{S}_1$.
  2. As the constituent Identity Based Signature scheme, S_{IBS}, is chosen by $\mathcal{B}$, so $\mathcal{B}$ generates the secret key $SK_{ID^{(i)}}^{IBS}$ corresponding to $ID^{(i)}$, then adds $(ID^{(i)}, SK_{ID^{(i)}}^{IBS})$ into the list $\mathcal{S}_2$.

  $\mathcal{B}$ finally returns $(SK_{ID^{(i)}}^{IBE}, SK_{ID^{(i)}}^{IBS})$ to $\mathcal{A}$.

- *Signcryption* queries : The response to the signcryption query for the message $m^{(i)}$ corresponding to the receiver's identity $ID_{Rec^{(i)}}$ and the sender's identity $ID_{Sen^{(i)}}$ is as follows :

  1. $\mathcal{B}$ searches the list $\mathcal{S}_2$ for the secret key corresponding to identity $ID_{Sen^{(i)}}$. If it does not exist, $\mathcal{B}$ generates the secret key corresponding to $ID_{Sen^{(i)}}$ using KeyGen^{IBS} algorithm of S_{IBS}. Let $SK_{ID^{(i)}}^{IBS}$ be the corresponding secret key.
  2. $\mathcal{B}$ then chooses a random number $r^{(i)}$ and runs $c'^{(i)} \leftarrow$ Encrypt$(r^{(i)}, ID_{Rec^{(i)}}, Params^{IBE})$.
  3. $\mathcal{B}$ searches the list L_1 for the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a string $h_1^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.

4. $\mathcal{B}$ runs $\sigma^{(i)} \leftarrow \text{Sign}(h^{(i)}, SK^{IBS}_{ID_{Sen^{(i)}}}, ID_{Sen^{(i)}}, Params^{IBS})$.
5. Then, $\mathcal{B}$ searches the list L_2 for the tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
6. $\mathcal{B}$ then computes $c^{(i)} = h_2^{(i)} \oplus m^{(i)}$.

$\mathcal{B}$ finally sends $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h^{(i)}, \sigma^{(i)})$ to $\mathcal{A}$.

- *Designcryption* queries : For input $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ and $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ (receiver's and sender's identities are $ID_{Rec^{(i)}}$ and $ID_{Sen^{(i)}}$ respectively), $\mathcal{B}$ responds as follows:
  1. $\mathcal{B}$ first runs $x \leftarrow \text{Verify}(h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, Params^{(i)})$. If x is false, return $\perp$, else
  2. $\mathcal{B}$ sends $(c'^{(i)}, ID_{Rec^{(i)}})$ to $\mathcal{CH}$ to decrypt. Let $r^{(i)}$ be the output from the Decrypt algorithm of S_{IBE}.
  3. Then $\mathcal{B}$ searches the list L_2 for tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_2^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_1.
  4. $\mathcal{B}$ then computes $m^{(i)} = h_2^{(i)} \oplus c^{(i)}$.
  5. Now, $\mathcal{B}$ searches the list L_1 for the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1'^{(i)})$. If such a tuple exists and $h_1'^{(i)} = h_1^{(i)}$, then $\mathcal{B}$ returns $(m^{(i)}, (r^{(i)}, h_1^{(i)}, \sigma^{(i)}))$, else $\perp$. If such a tuple does not exist, then $\mathcal{B}$ chooses a string $h_1'^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1'^{(i)})$ to the list L_1. If $h_1'^{(i)} = h_1^{(i)}$, then $\mathcal{B}$ returns $(m^{(i)}, (r^{(i)}, h_1^{(i)}, \sigma^{(i)}))$, else $\perp$.

Once $\mathcal{A}$ decides to enter the challenge phase, it chooses two messages m_0, m_1 of same length and two identities ID_R and ID_S corresponding to the receiver's and sender's identities respectively and sends them to $\mathcal{B}$ which is responded as follows:

1. $\mathcal{B}$ then chooses two random strings $r_0, r_1 \in \{0,1\}^{l_1}$ and the receiver's identity ID_R and sends them to $\mathcal{CH}$.
2. $\mathcal{CH}$ then chooses a bit, say b, uniformly at random from $\{0,1\}$ and computes the ciphertext $c'_b \leftarrow \text{Encrypt}(r_b, ID_R, Params^{IBE})$.
3. $\mathcal{CH}$ then sends the ciphertext c'_b to $\mathcal{B}$.
4. $\mathcal{B}$ then searches the list L_1 for the tuple $(m_0, r_0, ID_R, ID_S, h_{1,0})$. If such a tuple does not exist, $\mathcal{B}$ chooses a string $h_{1,0}$ uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m_0, r_0, ID_R, ID_S, h_{1,0})$.
5. $\mathcal{B}$ searches the list $\mathcal{S}_2$ for the secret key corresponding to the identity ID_S. If the secret key does not exist, $\mathcal{B}$ then runs KeyGen algorithm of S_{IBS}. Let the secret key be $SK^{IBS}_{ID_S}$.

6. After that $\mathcal{B}$ runs the Sign algorithm of S_{IBS} on $h_{1,0}$ and identity ID_S to get signature $\sigma_0 \leftarrow \text{Sign}(h_{1,0}, SK_{ID_S}^{IBS}, ID_S, Params^{IBS})$.

7. $\mathcal{B}$ then searches the tuple $(r_0, c'_b, h_{1,0}, \sigma_0, ID_R, ID_S, h_{2,0})$ in the list L_2. If such a tuple does not exist. $\mathcal{B}$ chooses a string uniformly at random, say $h_{2,0}$ from $\{0,1\}^{l_1}$ and adds the tuple $(r_0, c_b, h_{1,0}, \sigma_0, ID_R, ID_S, h_{2,0})$ to the list L_2.

8. $\mathcal{B}$ then computes $c_0 = h_{2,0} \oplus m_0$. Let $\mathcal{C}_0 \equiv (c_0, c'_b, h_{1,0}, \sigma_0)$.

9. $\mathcal{B}$ repeats the above 5 steps taking r_1 and m_1. Let $\mathcal{C}_1 \equiv (c_1, c'_b, h_{1,1}, \sigma_1)$.

Finally, $\mathcal{B}$ chooses a bit $v \in \{0,1\}$ uniformly at random and returns $\mathcal{C}_v$ to $\mathcal{A}$.

$\mathcal{A}$ then performs a second series of queries which is treated in the same way for H_1, H_2, Secret Key (except secret key query on identity ID_R) and *Signcryption* queries. For *Designcryption* queries, given an input $\mathcal{C}^{(j)} = (c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$ for $ID_{Rec^{(j)}}$ & $ID_{Sen^{(j)}}$, $\mathcal{B}$ responds as follows:

1. If $(c'^{(j)}, ID_{Rec^{(j)}}) \neq (c'_b, ID_R)$, $\mathcal{B}$ responds in the same way as it does for *Designcryption* queries before the challenge phase.

2. If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}) = (c'_b, h_{1,v}, \sigma_v, ID_R, ID_S)$, then we assume $c^{(j)} \neq c_v$ (otherwise $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$ will be the same as the challenge ciphertext for the corresponding receiver's identity $ID_{Rec^{(j)}}$ and sender's identity $ID_{Sen^{(j)}}$). Since $(c'_b, ID_R) = (c'^{(j)}, ID_{Rec^{(j)}})$, decryption of $c^{(j)}$ for identity $ID_{Rec^{(j)}}$ will yield $r^{(j)}$ which will be the same as r_b. $\mathcal{B}$ then searches the tuple $(r_0, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,0})$ in the list L_2 and computes $m_0^{(j)}$ by taking $\oplus$ of $c^{(j)}$ and $h_{2,0}$, i.e., $m_0^{(j)} = c^{(j)} \oplus h_{2,0}$. Similarly, $\mathcal{B}$ searches the tuple $(r_1, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,1})$ in the list L_2 and computes $m_1^{(j)} = c^{(j)} \oplus h_{2,1}$. $\mathcal{B}$ then searches the tuple $(m_0^{(j)}, r_0, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ and $(m_1^{(j)}, r_1, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ in the list L_1. If either of the tuples exists, then $\mathcal{B}$ aborts the game and returns the corresponding bit as the final guess bit to the challenger. If no such tuples exist, then $\mathcal{B}$ chooses two strings uniformly at random, say $h_1'^{(j)}$ and $\bar{h}_1'^{(j)}$ from $\{0,1\}^{l_1}$ such that $h_1'^{(j)} \neq h_1^{(j)}$ and $\bar{h}_1'^{(j)} \neq h_1^{(j)}$. $\mathcal{B}$ then adds the tuples $(m_0^{(j)}, r_0, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1'^{(j)})$ and $(m_1^{(j)}, r_1, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, \bar{h}_1'^{(j)})$ to the list L_1 and returns $\perp$ to the adversary $\mathcal{A}$.

3. Suppose $(c'^{(j)}, ID_{Rec^{(j)}}) = (c'_b, ID_R)$, but $(h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) \neq (h_{1,v}, \sigma_v, ID_S)$. As $(c'_b, ID_R) = (c'^{(j)}, ID_{Rec^{(j)}})$, decryption of $c'^{(j)}$ for identity $ID_{Rec^{(j)}}$ will yield $r^{(j)}$ which will be the same as r_b. $\mathcal{B}$ then searches the tuple $(r_0, c^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,0}^{(j)})$ in the list L_2. If such a tuple doesn't exist, then $\mathcal{B}$ chooses a string uniformly at random, say $h_{2,0}^{(j)} \in \{0,1\}^{l_1}$ and adds the tuple $(r_0, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,0}^{(j)})$ in the list L_2. $\mathcal{B}$ then computes $m_0^{(j)} = c_j \oplus h_{2,0}^{(j)}$. Similarly, $\mathcal{B}$ searches the tuple $(r_1, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,1}^{(j)})$ in the list L_2. If such a tuple doesn't exist, then $\mathcal{B}$ chooses a string uniformly at random, say $h_{2,1}^{(j)} \in \{0,1\}^{l_1}$ and adds the tuple $(r_1, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_{2,1}^{(j)})$ in the list L_2. $\mathcal{B}$ then computes $m_1^{(j)} = c_j \oplus$

$h_{2,1}^{(j)}$. $\mathcal{B}$ then searches the tuple $(m_0^{(j)}, r_0, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ and $(m_1^{(j)}, r_1, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ in the list L_1. If either of the tuples exists, then $\mathcal{B}$ aborts the game and returns the corresponding bit as the final guess bit to the challenger. If no such tuple exists, then $\mathcal{B}$ chooses two strings uniformly at random, say $h_1'^{(j)}$ and $\bar{h}_1'^{(j)}$ from $\{0,1\}^{l_1}$ such that $h_1'^{(j)} \neq h_1^{(j)}$ and $\bar{h}_1'^{(j)} \neq h_1^{(j)}$. $\mathcal{B}$ then adds the tuples $(m_0^{(j)}, r_0, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1'^{(j)})$ and $(m_1^{(j)}, r_1, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, \bar{h}_1'^{(j)})$ in the list L_1 and returns $\perp$ to the adversary $\mathcal{A}$.

In the above designcryption queries, $\mathcal{B}$ aborts the game if the following two cases arise:

1. $\mathcal{A}$ queries for $(m_0^{(j)}, r_0, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ or $(m_1^{(j)}, r_1, ID_{Rec^{(j)}}, ID_{Sen^{(j)}}, h_1^{(j)})$ randomly to H_1 oracle. Since, the choice of r_0 and r_1 by $\mathcal{B}$ is completely random from the adversary $\mathcal{A}$'s point of view, this case occurs with negligible probability.

2. $\mathcal{A}$ correctly decrypts c' corresponding to the receivers identity ID_R. Hence, in this case, the probability of winning the game by $\mathcal{B}$ will be 1.

Note that in the above game, $\mathcal{B}$ interacts with $\mathcal{CH}$ as in the real game. Secret key query for identity ID_R has not been asked by $\mathcal{A}$ to $\mathcal{B}$, hence by $\mathcal{B}$ to $\mathcal{CH}$. Besides it, $\mathcal{B}$ has not queried on the challenge ciphertext to $\mathcal{CH}$.

At the end of the simulation, $\mathcal{B}$ will use the bit guessed by $\mathcal{A}$ to guess the challenge bit with S_{IBE}. If $\mathcal{A}$ guesses $w \in \{0,1\}$, $\mathcal{B}$ will output the same bit *viz* w to $\mathcal{CH}$. We divide the analysis of the success probability of $\mathcal{B}$ into two cases:

- $\mathcal{B}$ does not abort the game.

    1. If $b = v$, the simulation is perfect and the ciphertext, $\mathcal{C}_v$, produced by $\mathcal{B}$ will be a valid ciphertext of the message m_v corresponding to ID_R (receiver's identity) and ID_S (sender's identity). Let
        - E_1 denote the event that $\mathcal{B}$ wins.
        - E_2 denote the event that $b = v$.
        - E_3 denote the event that $\mathcal{A}$ wins.
        - E_4 denote the event that $\mathcal{C}_v$ is a valid ciphertext.

        Then
        $$\Pr[E_1|E_2] = \Pr[E_3|E_4]$$
    2. Suppose $b \neq v$.
        (a) Suppose $\mathcal{A}$ recognizes $\mathcal{C}_v$ as an invalid ciphertext. In that case, $\mathcal{B}$ will guess the bit $b = \bar{v}$, i.e., the complement of v. In this case, $\mathcal{B}$ will guess correctly. Let
        - E_5 denote the event that $b \neq v$.
        - E_6 denote the event that $\mathcal{A}$ recognizes $\mathcal{C}_v$ as an invalid ciphertext.

Then

$$\Pr[E_6|E_2] = 0$$

and

$$\Pr[E_1|E_5 \cap E_6] = 1$$

Let

$$\Pr[E_6|E_5] = p$$

(b) Now assume that $\mathcal{A}$ doesn't recognize $\mathcal{C}_v$ as an invalid ciphertext. In this case, from $\mathcal{A}'s$ point of view, $\mathcal{C}_v$ will appear as a random ciphertext unless $\mathcal{A}$ queries H_2 on input (r_v, c'_b). Since, r_v was chosen uniformly at random from $\{0,1\}^{l_1}$ by $\mathcal{B}$, hence the probability that $\mathcal{A}$ will query H_2 on input (r_v, c'_b) is negligible. So, from $\mathcal{A}'s$ point of view, $\mathcal{C}_v$ will appear as a random ciphertext. Let,

- E_7 denote the event that $\mathcal{A}$ recognizes $\mathcal{C}_v$ as a random ciphertext.

Then

$$\Pr[E_7|E_5] = 1 - p$$

$$\Pr[E_1|E_5 \cap E_7] = \tfrac{1}{2}$$

Note that

- $E_5 = E_2^C$
- $\Pr[E_2] = \Pr[E_5] = 1/2$
- $(E_5 \cap E_6) \cup (E_5 \cap E_7) = E_5$
- $E_2 \cap (E_5 \cap E_6) = \phi$
- $E_2 \cap (E_5 \cap E_7) = \phi$
- $(E_5 \cap E_6) \cap (E_5 \cap E_7) = \phi$
- $\Pr[E_5 \cap E_6] = \Pr[E_6|E_5]\Pr[E_5] = \frac{1}{2}p$
- $\Pr[E_5 \cap E_7] = \Pr[E_7|E_5]\Pr[E_5] = \frac{1}{2}(1-p)$

Therefore,

$$\Pr[E_1] = \Pr[E_1|E_2]\Pr[E_2] + \Pr[E_1|E_5 \cap E_6]\Pr[E_5 \cap E_6] + \Pr[E_1|E_5 \cap E_7]\Pr[E_5 \cap E_7] \tag{4.1}$$

$\Rightarrow \Pr[E_1] = \frac{1}{2}\Pr[E_1|E_2] + \frac{1}{2}p + \frac{1}{2}.\frac{1}{2}(1-p) \geq \frac{1}{2}\Pr[E_1|E_2] + \frac{1}{4}$
Since,

$$\Pr[E_1|E_2] = \Pr[E_3|E_4] \tag{4.2}$$

$\Rightarrow \Pr[E_1] \geq \frac{1}{2}\Pr[E_3|E_4] + \frac{1}{4}$

- $\mathcal{B}$ aborts the game.
  Let
  - E_8 denotes the event that $\mathcal{A}$ correctly decrypts the ciphertext c'.

Then

$$\Pr[\mathcal{B} \text{ aborts the game}] = \Pr[E_8]. \tag{4.3}$$

In this case,

$$\Pr[\mathcal{B} \text{ wins}] = 1. \tag{4.4}$$

Using equations (3) and (4), we get
$\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{B} \text{ wins} \mid \mathcal{B} \text{ aborts the game}]\Pr[\mathcal{B} \text{ aborts the game}] + \Pr[\mathcal{B} \text{ wins} \mid \mathcal{B} \text{ doesn't abort the game}]\Pr[\mathcal{B} \text{ doesn't abort the game}]$
$\Rightarrow \Pr[\mathcal{B} \text{ wins}] = 1.\Pr[E_8] + \Pr[E_1](1 - \Pr[E_8])$
$\Rightarrow \Pr[\mathcal{B} \text{ wins}] = \Pr[E_1] + \Pr[E_8](1 - \Pr[E_1])$
$\Rightarrow \Pr[\mathcal{B} \text{ wins}] \geq \Pr[E_1]$

Hence,
Advantage of $\mathcal{B} = \mathcal{A}dv(\mathcal{B}) = |\Pr[\mathcal{B} \text{ wins}] - \frac{1}{2}|$
$\geq |(\Pr[E_1] - \frac{1}{2})| \geq \frac{1}{2}(|\Pr[E_3|E_4] - \frac{1}{2}|) = \frac{1}{2}(\mathcal{A}dv(\mathcal{A})) = \frac{\epsilon}{2}$

4.8.2 Ciphertext Unforgeability

We can similarly prove ciphertext unforgeability. We show that our scheme is ESUF-IBSC-CMA secure under the random oracle model provided the underlying IBS scheme is SUF-ID-CMA secure.

Theorem 9 *Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary which can break our scheme in the ESUF-IBSC-CMA game with an advantage ϵ in the random oracle model. Then there exists a PPT adversary $\mathcal{B}$ which can break S_{IBS} (Identity Based Signature scheme used) in the SUF-ID-CMA game with an advantage ϵ.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBS} algorithm of S_{IBS}. We shall show how to construct an SUF-ID-CMA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage ϵ against S_{IBS}. Suppose $\mathcal{B}$ receives public parameters $Params^{IBS}$ from $\mathcal{CH}$. $\mathcal{B}$ chooses an Identity Based Encryption scheme S_{IBE} whose public parameters $Params^{IBE}$ are independently generated from the public parameters of S_{IBS}. We can safely assume that $Params^{IBE} \cap Params^{IBS} = \phi$. $\mathcal{B}$ maintains lists L_1 and L_2 for queries on hash functions H_1, H_2. Besides these, $\mathcal{B}$ maintains two other lists $\mathcal{S}_1$ and $\mathcal{S}_2$ for queries on secret keys of different identities corresponding to S_{IBE} and S_{IBS} .

We now explain how requests from $\mathcal{A}$ are treated by $\mathcal{B}$. The response to H_1 and H_2 queries are exactly as in the proof of Theorem 8.

- *KeyGeneration* queries : For an input $ID^{(i)}$ from $\mathcal{A}$, algorithm $\mathcal{B}$ responds to $\mathcal{A}$ in two steps:

  1. $\mathcal{B}$ sends $ID^{(i)}$ to $\mathcal{CH}$. Let $\mathcal{CH}$ returns the corresponding secret key $SK^{IBS}_{ID^{(i)}}$. $\mathcal{B}$ then adds $(ID^{(i)}, SK^{IBS}_{ID^{(i)}})$ into the list $\mathcal{S}_2$.

2. As the constituent Identity Based Encryption scheme, S_{IBE}, is chosen by $\mathcal{B}$, so $\mathcal{B}$ generates the secret key $SK^{IBE}_{ID^{(i)}}$ corresponding to $ID^{(i)}$. $\mathcal{B}$ then adds $(ID^{(i)}, SK^{IBE}_{ID^{(i)}})$ into the list $\mathcal{S}_1$.

$\mathcal{B}$ finally returns $(SK^{IBE}_{ID^{(i)}}, SK^{IBS}_{ID^{(i)}})$ to $\mathcal{A}$.

- *Signcryption* queries : The response to signcryption query for message $m^{(i)}$ corresponding to the receiver's identity $ID_{Rec^{(i)}}$ and sender's identity $ID_{Sen^{(i)}}$ is as follows :

1. $\mathcal{B}$ first chooses a number, say $r^{(i)} \in \mathcal{R}$.
2. $\mathcal{B}$ then runs $c'^{(i)} \leftarrow \text{Encrypt}(r^{(i)}, ID_{Rec^{(i)}}, Params^{IBE})$.
3. $\mathcal{B}$ then checks the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$ in the list L_1. If such a tuple doesn't exist, then $\mathcal{B}$ chooses a string, say $h_1^{(i)}$,uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1^{(i)})$ to the list L_1.
4. $\mathcal{B}$ then sends $(h_1^{(i)}, ID_{Sen^{(i)}})$ to $\mathcal{CH}$. $\mathcal{CH}$ runs $\sigma^{(i)} \leftarrow \text{Sign}(h_1^{(i)}, SK^{IBS}_{ID_{Sen^{(i)}}})$ and sends $\sigma^{(i)}$ to $\mathcal{B}$.
5. Then, $\mathcal{B}$ searches the list L_2 for tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses a random string, say $h_2^{(i)}$, uniformly at random from $\{0,1\}^{l_2}$ and adds $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
6. $\mathcal{B}$ then computes $c^{(i)} = h_2^{(i)} \oplus m^{(i)}$.

$\mathcal{B}$ finally sends $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ to $\mathcal{A}$.

- *Designcryption* queries : For input $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ and $ID_{Rec^{(i)}}$, $ID_{Sen^{(i)}}$ (receiver's and sender's identities are $ID_{Rec^{(i)}}$ and $ID_{Sen^{(i)}}$ respectively), $\mathcal{B}$ responds as follows:

1. $\mathcal{B}$ searches the list $\mathcal{S}_1$ for the secret key corresponding to the identity $ID_{Rec^{(i)}}$. If it does not exist, $\mathcal{B}$ generates the secret key corresponding to $ID_{Rec^{(i)}}$ using the KeyGen^{IBE} algorithm of S_{IBE}. Let $SK^{IBE}_{ID_{Rec^{(i)}}}$ be the corresponding secret key.
2. $\mathcal{B}$ first runs $x \leftarrow \text{Verify}(h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, Params^{IBS})$. If x is false, return $\perp$, else
3. $\mathcal{B}$ runs $r^{(i)} \leftarrow \text{Decrypt}(c'^{(i)}, SK^{IBE}_{ID_{Rec^{(i)}}}.ID_{Rec^{(i)}}, Params^{IBE})$.
4. Then $\mathcal{B}$ searches the list L_2 for tuple $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$. If such a tuple does not exist, $\mathcal{B}$ chooses uniformly at random a string $h_2^{(i)}$ from $\{0,1\}^{l_1}$ and adds $(r^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_2^{(i)})$ to the list L_2.
5. $\mathcal{B}$ then computes $m^{(i)} = h_2^{(i)} \oplus c^{(i)}$.
6. Now, $\mathcal{B}$ searches the list L_1 for the tuple $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1'^{(i)})$. If such a tuple exists and $h_1'^{(i)} = h_1^{(i)}$, then $\mathcal{B}$ returns $(m^{(i)}, (r^{(i)}, h_1^{(i)}, \sigma^{(i)}))$ else $\perp$. If such

a tuple does not exist, then $\mathcal{B}$ chooses a string $h_1'^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds $(m^{(i)}, r^{(i)}, ID_{Rec^{(i)}}, ID_{Sen^{(i)}}, h_1'^{(i)})$ to the list L_1. If $h_1'^{(i)} = h_1^{(i)}$, then $\mathcal{B}$ returns $(m^{(i)}, (r^{(i)}, h_1^{(i)}, \sigma^{(i)}))$, else $\perp$.

Once this game is over, $\mathcal{A}$ submits a ciphertext $\mathcal{C} = (c, c', h_1, \sigma)$ corresponding to the receiver's identity ID_R and the sender's identity ID_S such that the secret key corresponding to ID_S has not been queried earlier. $\mathcal{B}$ then submits (h_1, σ) to $\mathcal{CH}$. Regarding (h_1, σ, ID_S), the following cases arise:

1. $\mathcal{B}$ has not queried to $\mathcal{CH}$ on h_1 corresponding to the sender's identity ID_S. In this case, $\mathcal{B}$ wins the game.

2. $\mathcal{B}$ has queried to $\mathcal{CH}$ on h_1 corresponding to the sender's identity ID_S. Hence, $\mathcal{A}$ has obtained some ciphertext $\mathcal{C}_i = (c_i, c_i', h_{1i}, \sigma_i)$ after *Signcryption* query over some receiver's identity ID_{Rec_i} and sender's identity ID_{Sen_i} where $(h_{1i}, ID_{Sen_i}) = (h_1, ID_S)$. If $\sigma_i \neq \sigma$, $\mathcal{B}$ wins the game.

3. (h_1, σ) corresponding to the sender's identity has been obtained by $\mathcal{B}$ from the $\mathcal{CH}$; hence, $\mathcal{A}$ has obtained some ciphertext $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ from the Signcryption oracle for some receiver's identity $ID_{Rec^{(i)}}$ and $ID_{Sen^{(i)}}$. Let $\sigma^{(i)} = \sigma$. Then $(h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}) = (h_1, \sigma, ID_S)$. Since $h_1^{(i)} = h_1$, with negligible probability, two different input value on H_1 will yield the same output (assuming H_1 as a random function). Hence, designcryption of $\mathcal{C} = (c, c', h_1, \sigma)$, say m, corresponding to the receiver's identity ID_R and the sender's identity ID_S, and the designcryption of $\mathcal{C}^{(i)} = (c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$, say $m^{(i)}$, corresponding to the receiver's identity $ID_{Rec^{(i)}}$ and sender's identity $ID_{Sen^{(i)}}$ will be same, i.e. $m = m^{(i)}$. Again decryption of c', say r, corresponding to the receiver's identity ID_R and decryption of $c'^{(i)}$, say $r^{(i)}$, corresponding to the receiver's identity $ID_{Rec^{(i)}}$ will also be the same, i.e. $r = r^{(i)}$. Moreover, $ID_R = ID_{Rec^{(i)}}$. Hence in this case, $(m, r, h^{(i)}, \sigma) = (m^{(i)}, r^{(i)}, h_1^{(i)}, \sigma^{(i)})$ corresponding to the sender's identity ID_S. So, this case will not arise as per the definition of ESUF-IBSC-CMA's security game.

Case 1 and 2 implies that whenever $\mathcal{C} = (c, c', h_1, \sigma)$ is a valid forged ciphertext submitted by $\mathcal{A}$ defined in the security game of ESUF-IBSC-CMA, the forged message-signature pair (h_1, σ) submitted by $\mathcal{B}$ to $\mathcal{CH}$ will be a valid message-signature pair according to the definition of security game of SUF-ID-CMA.

Hence, whenever $\mathcal{A}$ is able to submit a valid ciphertext to $\mathcal{B}$, $\mathcal{B}$ will also be able to submit a valid forged message-signature pair to $\mathcal{CH}$.

So, advantage of $\mathcal{B} = \mathcal{A}dv(\mathcal{B}) = \Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{A} \text{ wins}] = \mathcal{A}dv(\mathcal{A}) = \epsilon$.

4.9 Efficiency

1. Time Efficiency : In our proposed scheme, Encrypt and Sign can be done in parallel in the Signcrypt algorithm and Decrypt and Verify can be done in parallel in the Designcrypt algorithm. Let t_E, t_S, t_D, t_V and t_H be the time taken by the Encrypt, Sign, Decrypt, Verify and Hash algorithms respectively. Then, assuming that the Sign and Encrypt are run concurrently in Signcrypt, the time taken by our scheme in Signcrypt will be $T_{SC} = \max(t_E, t_S) + 2t_H$ and assuming that Decrypt and Verify are run concurrently in Designcrypt, the time taken by our scheme in Designcrypt will be $T_{DSC} = \max(t_D, t_V) + 2t_H$; whereas in the Sign-then-Encrypt approach, the total time taken in signcryption will be $(t_E + t_S)$ and in designcryption will be $(t_D + t_V)$. In general, $t_H << (t_E$ or t_S or t_D or $t_V)$.

2. Space Efficiency : In many cases, in practice, the ciphertext length bears (approx.) a constant ratio with the plaintext. This is also the case with many signature schemes. Let the output length of the Encrypt algorithm be (at most) αl_1, where l_1 is the bit length of message m. Let the output length of the signature corresponding to an l_1 bit message be (at most) $l_2 = \beta l_1$. Hence, the total length of ciphertext will be (at most) $(\alpha + \beta + 2)l_1$. But in the Sign-then-Encrypt approach, ciphertext length will be, roughly, $\alpha(\beta + 1)l_1$. Hence, our scheme is likely to produce a shorter ciphertext length compared to the Sign-then-Encrypt approach if $\alpha \geq \frac{2}{\beta} + 1$.

4.10 Comparisons

Using our generic method, we composed two IBSC scheme - first one by composing Boneh-Franklin Identity Based Encryption (BF-IBE) [15] with Shamir's Identity Based Signature (SH-IBS) [85] scheme and the second one by composing Boneh-Franklin IBE [15] with Kurosawa-Heng Identity Based Signature (KH-IBS) ([59], page 113 of [63]) scheme. We compared these schemes with the Identity Based Signcryption (IBSC) schemes proposed by Boyen [18], Chen-Malone-Lee [25] and Barreto et. al. [7]. BF-IBE + SH-IBS (Boneh-Franklin IBE and Shamir IBS) has more than double ciphertext overhead and BF-IBE + KH-IBS (Boneh-Franklin IBE and Kurosawa-Heng IBS) has almost double ciphertext overhead than IBSC schemes proposed by Boyen, Chen-Malone-Lee and Barreto. In case of time efficiency, both schemes (BF-IBE + SH-IBS and BF-IBE + KH-IBS) take less time in Signcrypt and Designcrypt (note to remember that in our method, in Signcrypt, Encrypt and Sign algorithm can be run in parallel and in Designcrypt, Decrypt and Verify can be run in parallel) compared to Boyen and Chen-Malone-Lee IBSC scheme. Barreto's scheme has lower cost of computation in Signcrypt than that of BF-IBE + SH-IBS and BF-IBE + KH-IBS but in Designcrypt, both schemes, BF-IBE + SH-IBS has lower cost of computation than that of Barreto. Summary of the efficiency comparisons has been given in table 4.5.

Table 4.5: Efficiency Comparisons

Scheme	Signcrypt				Designcrypt			Ciphertext Length
	E	P	SM		E	P	SM	
Boyen IBSC [18]	1	1	3		0	4	2	$2\|G_1\| + \|ID\| + \|M\|$
Chen-Malone-Lee IBSC [25]	0	1	3		0	3	1	$2\|G_1\| + \|ID\| + \|M\|$
Barreto et. al. IBSC [7]	1	0	3		1	2	1	$2\|G_1\| + \|M\|$
Different IBSC constructed using IBSC-Scheme1 generic method								
BF-IBE [15] + SH-IBS [85]	1	1	1		2	1	1	$\|G_1\| + 2\|\mathbb{Z}_N^*\| + 3\|M\|$
BF-IBE [15] + KH-IBS ([59], [63])	1	1	2		1	2	1	$2\|G_1\| + \|\mathbb{Z}_q^*\| + 3\|M\|$
Different IBSC constructed using IBSC-Scheme2 generic method								
BF-IBE [15] + SH-IBS [85]	1	1	1		0	1	1	$\|G_1\| + 2\|\mathbb{Z}_N^*\| + 4\|M\|$
BF-IBE [15] + KH-IBS ([59], [63])	1	1	2		0	1	1	$2\|G_1\| + \|\mathbb{Z}_q^*\| + 4\|M\|$

- E denotes number of exponentiation.
- P denotes number of pairing. We assume $e : G_1 \times G_1 \to G_T$, if e is a symmetric bilinear map, else $e : G_1 \times G_2 \to G_T$, in case of asymmetric bilinear map.
- SM denotes number of scalar multiplication of a point on elliptic curve.
- $|G_1|$ denotes the bit length of an element in group G_1 used in pairing.
- $|G_2|$ denotes the bit length of an element in group G_2 used in pairing.
- $|G_T|$ denotes the bit length of an element in group G_T used in pairing.
- $|ID|$ denotes the bit length of an identity.
- $|M|$ denotes the message length.
- $|Z_N^*|$ denotes the length of an element in Z_N^* where $N = pq$ is the product of two prime numbers p and q.
- $|Z_q^*|$ denotes the length of an element of Z_q^*. Here q is a prime number.
- BF-IBE denotes Boneh-Franklin Identity Based Encryption scheme [15].
- SH-IBS denotes Shamir Identity Based Signature scheme [85].
- KH-IBS denotes Kurosawa-Heng Identity Based Signature scheme ([59],[63]).

Chapter 5

Achieving CCA-secure Signcryption Schemes from OWE-secure Encryption Schemes

In this chapter, we show how to obtain a signcryption scheme that is dM-IND-iCCA secure even when the underlying Encryption scheme is One-Way secure or One-Way secure against plaintext checking attack. We have proposed two signcryption schemes, *Scheme 1* and *Scheme 2*. *Scheme 1* achieves dM-IND-iCCA security when the underlying encryption scheme is One-Way secure while *Scheme 2* achieves dM-IND-iCCA security when the underlying encryption scheme is One-Way secure against plaintext checking attack. The transformation made to construct *Scheme 1* is reminiscent to Fujisaki-Okamoto transformation.

5.1 Introduction

Confidentiality and Authentication are two of the major goals in cryptography. One obvious way to achieve both is to use encryption and signature as black boxes. Some of the ways by which it is achieved are Encrypt and Sign, Encrypt then Sign and Sign then Encrypt. It can be easily shown that the first two are not secure since they do not achieve confidentiality (not indistinguishable) in two-user and multi-user setting respectively. Sign then Encrypt approach achieves both confidentiality and unforgeability but at the expense of higher computational cost and ciphertext overhead.

The simplest security model for a signcryption scheme considers the two two-user setting [3, 34] in which the interaction takes place between sender and receiver only. But this setting does not incorporate the real setting, as pointed out by [33], in which multiple senders may interact with the same receiver and multiple receivers may interact with the same sender. Hence, a more realistic setting is multi-user setting. Security of signcryption schemes can be further seen into two different attack models, namely *insider* and *outsider* attack model. In the outsider security model, adversary is not allowed to possess the secret keys of the sender and receiver but in the insider security model, attacker may possess the secret keys

of one of the parties. Currently, the strongest security model is insider attack model in the multi-user setting. Libert et. al. [64] proposed first fully insider secure signcryption scheme in the multi-user setting. Now, there are several schemes [65, 64, 33] which achieve the strongest security in the random oracle model [11]. There are several schemes [3, 88, 69] which are secure secure in the standard model but does not achieve the strongest security notion. In 2011, Chiba-Matsuda-Schuldt-Matsuura [26] have given a generic construction of signcryption scheme which achieves the strongest security (insider security in the multi-user setting) in the standard model. They used Sign-then-Encrypt paradigm which relies upon chosen-ciphertext-secure tag-based KEM, a chosen-ciphertext-secure DEM that has a one-to-one property, and a strongly unforgeable signature scheme.

Earlier, An, Dodis and Rabin [3] have proposed a Commit-then-Encrypt& Sign paradigm ($\mathcal{C}t\mathcal{E}\&\mathcal{S}$) to construct a signcryption scheme. The main feature of this approach is that both Encryption and Signature can be done in parallel in the Signcryption phase and Decryption and Verification can be done in parallel in the Designcryption phase. But from the point of security, this scheme achieves generalized indistinguishability against chosen ciphertext attack if *Commit* has the hiding property and *Encryption* is (generalized) IND-CCA secure(a slightly weaker notion); while it is unforgeable against chosen message attacks if *Commit* achieves the relaxed binding property and *Signature* is EUF-CMA secure i.e. existentially unforgeable under chosen message attacks.

In this chapter, we have proposed two schemes for building secure signcryption schemes. In the first scheme, Scheme 1, an encryption scheme which is OWE-CPA secure and a signature scheme which is EUF-CMA secure yield dM-IND-iCCA and dM-EUF-iCMA secure signcryption scheme in the random oracle model whereas in the second scheme, Scheme 2, an encryption scheme which is OWE-PCA secure and a signature scheme which is EUF-CMA secure yield dM-IND-iCCA and dM-EUF-iCMA secure signcryption scheme again in the random oracle model. Scheme 1 and Scheme 2 do not just only provide ways to construct signcryption schemes but also it shows how to construct a highly secure signcryption scheme from the less secure parent encryption schemes. The construction of Scheme 1 is familiar to Fujisaki-Okamoto transformation [43].

5.2 Proposed Scheme : Scheme 1

Let $\mathcal{E} = (\mathrm{KG}^E, \mathrm{ENC}, \mathrm{DEC})$ and $\mathcal{S} = (\mathrm{KG}^S, \mathrm{SIG}, \mathrm{VER})$ be encryption and signature schemes respectively. If the encryption algorithm ENC is supplied with a random string r, externally, we denote it by ENC_r, else by ENC. Let l_1 be the bit-length of the signing message used in the algorithm SIG of $\mathcal{S}$. Let l_2 be the bit-length of any message m from the message space $\mathbf{M}$. Moreover, let l_3 be the bit-length of a random number r from the random space $\mathbf{R}$.

The construction of Scheme 1 is described in Table 5.1.

Scheme 1	
Setup(1^λ) • Choose two cryptographically secure hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $H_2 : \{0,1\}^* \to \{0,1\}^{l_2+l_3}$ where l_1 = length of the signing message used in the signature scheme. l_2 = length of the message m. l_3 = length of the random string used in the encryption scheme. $Params \leftarrow (H_1, H_2, 1^\lambda)$	**SC**$(m, PK_{Rec}, PK_{Sen}, SK_{Sen}, Params)$ • $r, r' \leftarrow \mathbf{R}$ • $h_1 = H_1(m, r, r', ID_{Rec})$ • $c' \leftarrow \text{ENC}_r(r', PK^E_{Rec})$ • $\sigma \leftarrow \text{SIG}(h_1, SK^S_{Sen}, PK^S_{Sen})$ • $h_2 = H_2(r', c', h_1, \sigma, ID_{Sen})$ • $c = h_2 \oplus m \| r$ Ciphertext $\mathcal{C} \equiv (c, c', h_1, \sigma)$
KG$_{Rec}(Params)$ **&** **KG**$_{Sen}(Params)$ • $(PK^E, SK^E) \leftarrow \text{KG}^E(1^\lambda)$ • $(PK^S, SK^S) \leftarrow \text{KG}^S(1^\lambda)$ • $PK_{Rec} = (PK^E_{Rec}, PK^S_{Rec})$ • $SK_{Rec} = (SK^E_{Rec}, SK^S_{Rec})$ • $PK_{Sen} = (PK^E_{Sen}, PK^S_{Sen})$ • $SK_{Sen} = (SK^E_{Sen}, SK^S_{Sen})$	**DSC**$(\mathcal{C} \equiv (c, c', h_1, \sigma), SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$ • $r' \leftarrow \text{DEC}(c', SK^E_{Rec}, PK^E_{Rec})$ • $x \leftarrow \text{VER}(h_1, \sigma, PK^S_{Sen})$. If r' is $\perp$ or x is false, return $\perp$; else go to the next step. • $h_2 = H_2(r', c', h_1, \sigma, ID_{Sen})$ • $m \| r = h_2 \oplus c$ • if $h_1 \neq H_1(m, r, r', ID_{Rec})$, return $\perp$, else go to next step • if $\text{ENC}_r(r', PK^E_{Rec}) = c'$, return m, else $\perp$.

Table 5.1: Construction of Scheme 1

5.3 Security of Scheme 1

5.3.1 Confidentiality

Theorem 10 *In the random oracle model, if there exists a dM-IND-iCCA adversary $\mathcal{A}$ which can distinguish ciphertexts during the relevant game with a non-negligible advantage, then there exists an algorithm $\mathcal{B}$ which can break the OWE security of the parent encryption scheme $\mathcal{E}$ with a non-negligible advantage. Formally,*

$$Pr[\mathcal{B}\ wins] = \mathcal{A}dv(\mathcal{B}) \geq \begin{cases} 2\mathcal{A}dv(\mathcal{A}) & \text{if } q_{H_2} = 0, \\ \dfrac{2\mathcal{A}dv(\mathcal{A})}{q_{H_2}} & \text{otherwise;} \end{cases}$$

where q_{H_2} is the number of H_2 oracle queries.

Proof : We shall show how to construct an OWE adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain a non-negligible advantage against $\mathcal{E}$.

Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup algorithm of Scheme 1. For fixed receiver, *Rec*, suppose algorithm $\mathcal{B}$ receives the public key, PK^E_{Rec}, of the encryption scheme $\mathcal{E}$ from $\mathcal{CH}$. As the proof is in the insider security model, we further assume that signature scheme(s) is(are) known to $\mathcal{A}$ and public key-secret key of signature scheme(s) for all user(s) is(are) also known to $\mathcal{A}$. $\mathcal{B}$ then sends PK^E_{Rec} to the adversary $\mathcal{A}$.

$\mathcal{B}$ maintains three lists $\mathcal{L}_{H_1}$, $\mathcal{L}_{H_2}$ and $\mathcal{L}^{In}_{Dec}$ to maintain the consistency while replying to the query asked by the adversary $\mathcal{A}$. $\mathcal{L}_{H_1}$ contains the entries of the form $(m^{(j)}, r^{(j)}, r'^{(j)}, ID_{Rec}, h_1^{(j)})$ in response to the H_1 queries. $\mathcal{L}_{H_2}$ contains the entries of the form $(r'^{(k)}, c'^{(k)}, h_1^{(k)}, \sigma^{(k)}, h_2^{(k)}, ID_{Sen^{(k)}})$ in response to the H_2 queries and $\mathcal{L}^{In}_{Dec}$ contains the entries of invalid ciphertexts of the form $(c^{(l)}, c'^{(l)}, h_1^{(l)}, \sigma^{(l)}, ID_{Rec}, ID_{Sen^{(l)}})$ which have been responded by $\mathcal{B}$ after querying by $\mathcal{A}$ to the *Designcryption* oracle. $\mathcal{B}$ additionally maintains a temporary list $\mathcal{L}_T$ to store different entries used at intermediate step while responding to the different queries made by $\mathcal{A}$. Initially, all lists are empty.

First query can be either on H_1 or H_2 or *Designcryption*.

- For H_1 query of the form $(m^{(1)}, r^{(1)}, r'^{(1)}, ID_{Rec})$, $\mathcal{B}$ chooses a string $h_1^{(1)}$ uniformly at random from $\{0,1\}^{l_1}$ and adds the tuple $(m^{(1)}, r^{(1)}, r'^{(1)}, ID_{Rec}, h_1^{(1)})$ into $\mathcal{L}_{H_1}$. $\mathcal{B}$ then returns $h_1^{(1)}$ to $\mathcal{A}$.

- For H_2 query of the form $(r'^{(1)}, c'^{(1)}, h_1^{(1)}, \sigma^{(1)}, ID_{Sen^{(1)}})$, $\mathcal{B}$ chooses a string $h_2^{(1)}$ uniformly at random from $\{0,1\}^{l_2}$ and adds the tuple $(r'^{(1)}, c'^{(1)}, h_1^{(1)}, \sigma^{(1)}, ID_{Sen^{(1)}}, h_2^{(1)})$ into $\mathcal{L}_{H_2}$. $\mathcal{B}$ then returns $h_2^{(1)}$ to $\mathcal{A}$.

- For *Designcryption* query of the form $(c^{(1)}, c'^{(1)}, h_1^{(1)}, \sigma^{(1)})$ for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(1)}}$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ and then adds the entry of the form $(c^{(1)}, c'^{(1)}, h_1^{(1)}, \sigma^{(1)}, ID_{Rec}, ID_{Sen^{(1)}})$ into $\mathcal{L}^{In}_{Dec}$.

We now explain how $\mathcal{B}$ responds to various queries made by $\mathcal{A}$ before the challenge phase. Suppose, till the $(i-1)^{\text{th}}$ queries, $\mathcal{B}$ responds in the following manner:

1. H_1 query: For any query of the form $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec})$ where $2 \leq p \leq (i-1)$, $\mathcal{B}$ searches in the list $\mathcal{L}_{H_1}$ for entry of the form $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec}, h_1^{(p)})$. If such an entry exists, $\mathcal{B}$ returns $h_1^{(p)}$ to $\mathcal{A}$. If such entry does not exist, $\mathcal{B}$ returns a string $h_1^{(p)}$ such that (a) it looks like a random string from $\mathcal{A}$'s point of view, (b) the previously declared invalid ciphertexts remain invalid and (c) the consistency of valid ciphertexts is preserved. Finally, $\mathcal{B}$ adds the entry $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec}, h_1^{(p)})$ into the list $\mathcal{L}_{H_1}$.

2. H_2 query: In a similar manner, for any query of the form $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}})$ where $2 \leq p \leq (i-1)$, $\mathcal{B}$ searches in the list $\mathcal{L}_{H_2}$ for entry of the form $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$. If such an entry exists, $\mathcal{B}$ returns $h_2^{(p)}$ to $\mathcal{A}$. If such an entry does not exist, $\mathcal{B}$ returns a string $h_2^{(p)}$ such that (a) it looks like a random string from $\mathcal{A}$'s point of view, (b) the previously declared invalid ciphertexts remain invalid and (c) the consistency of valid ciphertexts is preserved. Finally, $\mathcal{B}$ adds the entry $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$ into the list $\mathcal{L}_{H_2}$.

3. *Designcryption* query: For any query of the form $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$, for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(p)}}$ where $2 \leq p \leq (i-1)$, $\mathcal{B}$ returns $m^{(p)}$ if and only if there exist entries of the form $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec}, h_1^{(p)})$ in the list $\mathcal{L}_{H_1}$ and $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$ in the list $\mathcal{L}_{H_2}$ such that

   - $\text{VER}(h_1^{(p)}, \sigma^{(p)}, PK^S_{Sen^{(p)}}) = \text{true}$, and
   - $m^{(p)}||r^{(p)} = c^{(p)} \oplus h_2^{(p)}$, and
   - $\text{ENC}_{r^{(p)}}(r'^{(p)}, PK^E_{Rec}) = c'^{(p)}$.

   If $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$ is declared an invalid ciphertext, i.e., $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ for queried ciphertext $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$, $\mathcal{B}$ then adds the entry of the form $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Rec}, ID_{Sen^{(p)}})$ into the list $\mathcal{L}^{In}_{Dec}$.

Claim 1: Whenever $\mathcal{B}$ returns $m^{(p)}$ in the simulated game in response to a *Designcryption* query on the ciphertext $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$ by $\mathcal{A}$ on receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(p)}}$, Designcrypt algorithm returns the same message $m^{(p)}$ in the real game also. Moreover, whenever $\mathcal{B}$ returns $\perp$ in the simulated game, queried ciphertext is either invalid or $\mathcal{A}$ cannot prove the validity of the ciphertext anyhow.

Proof of Claim 1: It can be easily verified that in the real game also, Designcrypt algorithm returns the same message $(m^{(p)})$ whenever $\mathcal{B}$ returns $m^{(p)}$ for any queried ciphertext $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$ for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(p)}}$ in this simulated game.

Now, we show that in this simulated game, if $\mathcal{B}$ returns $\perp$, then either the queried ciphertext is indeed an invalid ciphertext or the adversary $\mathcal{A}$ cannot prove the validity of the ciphertext in any way whatsoever. If $\text{VER}(h_1^{(p)}, \sigma^{(p)}, PK^S_{Sen^{(p)}}) = \text{false}$, then the queried ciphertext is an invalid ciphertext. Else, let $\text{DEC}(c'^{(p)}, SK^E_{Rec}, PK^E_{Rec}) = r'^{(p)}$. $\mathcal{B}$ then searches the list $\mathcal{L}_{H_2}$ for entries of the form $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$. If such an entry does not exist, $\mathcal{B}$ returns $\perp$. In this case, $\mathcal{A}$ cannot prove the validity of the ciphertext as $\mathcal{B}$ can choose a random string $h_2^{(p)}$ in such a way that $h_2^{(p)} \oplus c^{(p)} = m^{(p)}||r^{(p)}$ and $\text{ENC}_{r^{(p)}}(r'^{(p)}, PK^E_{Rec}) \neq c'^{(p)}$. Else, if $(r'^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$ exists in $\mathcal{L}_{H_2}$, then $\mathcal{B}$ computes $h_2^{(p)} \oplus c^{(p)}$. Let $h_2^{(p)} \oplus c^{(p)} = m^{(p)}||r^{(p)}$. If $\text{ENC}_{r^{(p)}}(r'^{(p)}, PK^E_{Rec}) \neq c'^{(p)}$, $\mathcal{B}$ returns $\perp$. In this case, the queried ciphertext is indeed an invalid ciphertext. Else, if $\text{ENC}_{r^{(p)}}(r'^{(p)}, PK^E_{Rec}) = c'^{(p)}$, then $\mathcal{B}$ searches the entry of the form $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec}, h_1^{(p)})$. If such an entry does not exist, $\mathcal{B}$ returns $\perp$. In this case also, $\mathcal{A}$ cannot prove the validity of the ciphertext as $\mathcal{B}$ can choose a random string $\bar{h}_1^{(p)}$ such that $\bar{h}_1^{(p)} \neq h_1^{(p)}$. Hence, unless all of the above conditions satisfy, either the queried ciphertext is an invalid ciphertext or $\mathcal{A}$ cannot prove the validity of the ciphertext. $\square$

Now, we explain how $\mathcal{B}$ responds to $\mathcal{A}$ in the i^{th} query. The i^{th} query can be either H_1 or H_2 or *Designcryption* query.

H_1 **query:** For any query of the form $(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec})$, $\mathcal{B}$ follows Algorithm 1:

Algorithm 1

1. $\mathcal{B}$ searches in the list $\mathcal{L}_{H_1}$ for entries of the form $(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$. If it exists, $\mathcal{B}$ returns $h_1^{(i)}$ to $\mathcal{A}$, else go to next step.

2. $\mathcal{B}$ empties the list $\mathcal{L}_T$.

3. $\mathcal{B}$ then searches the list $\mathcal{L}_{H_2}$ for entries of the form $(r'^{(i)}, c'^{(i_n)}, h_1^{(i_n)}, \sigma^{(i_n)}, ID_{Sen^{(i_n)}}, h_2^{(i_n)})$. If such an entry exists, go to the next step, else $\mathcal{B}$ chooses a string $h_1^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and returns $h_1^{(i)}$ to the adversary $\mathcal{A}$. $\mathcal{B}$ then adds the entry of the form $(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$ into the list $\mathcal{L}_{H_1}$.

4. For each entry of the form $(r'^{(i)}, c'^{(i_n)}, h_1^{(i_n)}, \sigma^{(i_n)}, ID_{Sen^{(i_n)}}, h_2^{(i_n)})$, $\mathcal{B}$ searches the list $\mathcal{L}_{Dec}^{In}$ for entries of the form $(c^{(i_{n_t})}, c'^{(i_n)}, h_1^{(i_n)}, \sigma^{(i_n)}, ID_{Rec}, ID_{Sen^{(i_n)}})$. If such an entry exists, go to the next step; else $\mathcal{B}$ chooses a random string $h_1^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ and returns $h_1^{(i)}$ to the adversary $\mathcal{A}$. $\mathcal{B}$ then adds the entry of the form $(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$ into the list $\mathcal{L}_{H_1}$.

5. For each entry of the form $(c^{(i_{n_t})}, c'^{(i_n)}, h_1^{(i_n)}, \sigma^{(i_n)}, ID_{Rec}, ID_{Sen^{(i_n)}})$, $\mathcal{B}$ computes $m^{(i_{n_t})}||r^{(i_{n_t})} = c^{(i_{n_t})} \oplus h_2^{(i_n)}$. $\mathcal{B}$ then checks $\text{VER}(h_1^{(i_n)}, \sigma^{(i_n)}, PK_{Sen^{(i_n)}}^{S}) \stackrel{?}{=} \text{true}$, $\text{ENC}_{r^{(i)}}(r'^{(i)}, PK_{Rec}^{E}) \stackrel{?}{=} c'^{(i_n)}$, $m^{(i_{n_t})} \stackrel{?}{=} m^{(i)}$ and $r^{(i_{n_t})} \stackrel{?}{=} r^{(i)}$. If all four hold, then $\mathcal{B}$ adds $h_1^{(i_n)}$ into the list $\mathcal{L}_T$.

6. $\mathcal{B}$ finally chooses a string $h_1^{(i)}$ uniformly at random from $\{0,1\}^{l_1}$ such that $h_1^{(i)} \notin \mathcal{L}_T$. $\mathcal{B}$ then returns $h_1^{(i)}$ to the adversary $\mathcal{A}$ and adds the entry of the form $(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$ into the list $\mathcal{L}_{H_1}$.

Claim 2: The way $\mathcal{B}$ responds to the H_1 query made by $\mathcal{A}$, it is ensured that the string, $h_1^{(i)}$, returned by $\mathcal{B}$ looks like a random string from $\mathcal{A}$'s point of view. It is also ensured that previously declared invalid ciphertexts remain invalid. Moreover, consistency of previously queried ciphertext is also maintained.

Proof of Claim 2: It is easy to see from Algorithm 1 that the response given by $\mathcal{B}$ appears random from $\mathcal{A}$'s point of view.

Now, we show that previously declared invalid ciphertexts remain invalid. Let $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Rec}, ID_{Sen^{(p)}}) \in \mathcal{L}_{Dec}^{In}$, where $p < i$, be the p^{th} queried ciphertext to *Designcryption* oracle. If $\text{VER}(h_1^{(p)}, \sigma^{(p)}, PK_{Sen^{(p)}}^{S}) = \text{false}$, ciphertext remains invalid (Claim 1). If not, let $r'^{(p)} = \text{DEC}(c'^{(p)}, SK_{Rec}^{E}, PK_{Rec}^{E})$. If $r'^{(p)} \neq r'^{(i)}$, then the response from $\mathcal{B}$ does not affect the invalidity of the ciphertext $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$ (Claim 1). If $r'^{(p)} = r'^{(i)}$, then $\mathcal{B}$ searches in the list L_{H_2} for the entry $(r'^{(i)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)}, ID_{Sen^{(p)}}, h_2^{(p)})$ (Algorithm 1, step (3)). Again, the absence of such entry does not affect the invalidity of the ciphertext (Claim 1). If such

entry exists, $\mathcal{B}$ computes $m^{(p)}||r^{(p)} = c^{(p)} \oplus h_2^{(p)}$. $\mathcal{B}$ then checks $\text{ENC}_{r^{(i)}}(r'^{(i)}, PK_{Rec}^E) \stackrel{?}{=} c'^{(p)}$, $m^{(p)} \stackrel{?}{=} m^{(i)}$, $r^{(p)} \stackrel{?}{=} r^{(i)}$ (Algorithm 1, step (5)). If any one of conditions fails, the response from $\mathcal{B}$ does not affect the invalidity of the ciphertext (Claim 1). If all conditions satisfy, $\mathcal{B}$ then chooses a string $h_1^{(i)}$ such that $h_1^{(i)} \neq h_1^{(p)}$ (Algorithm 1, step (6)) preserving the invalidity of the previously declared invalid ciphertexts (Claim 1).

Now, we show that the way $\mathcal{B}$ responds, the consistency of previously queried valid ciphertexts is also maintained. Let $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$, where $p < i$, be the p^{th} queried valid ciphertext. Let the Designcrypt of the queried ciphertext outputs $m^{(p)}$. Then, there must be previously queried entry of the form $(m^{(p)}, r^{(p)}, r'^{(p)}, ID_{Rec}, h_1^{(p)})$ in $\mathcal{L}_{H_1}$, where $r'^{(p)} = \text{DEC}(c'^{(p)}, SK_{Rec}^E, PK_{Rec}^E)$ and $\text{ENC}_{r^{(p)}}(r'^{(p)}, PK_{Rec}^E) = c'^{(p)}$. Hence, if $(m^{(p)}, r^{(p)}, r'^{(p)}) = (m^{(i)}, r^{(i)}, r'^{(i)})$, then $\mathcal{B}$ returns $h_1^{(p)}$ which preserves the consistency of the previous queried valid ciphertext (Algorithm 1, step (1)), else if, $(m^{(p)}, r^{(p)}, r'^{(p)}) \neq (m^{(i)}, r^{(i)}, r'^{(i)})$, then the returned value $h_1^{(i)}$ from $\mathcal{B}$ does not affect the validity of the ciphertext $(c^{(p)}, c'^{(p)}, h_1^{(p)}, \sigma^{(p)})$. □

Now, we show how $\mathcal{B}$ responds when the i^{th} query is on H_2.

H_2 **query:** For any query of the form $(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}})$, $\mathcal{B}$ follows Algorithm 2:

Algorithm 2

1. $\mathcal{B}$ searches in the list $\mathcal{L}_{H_2}$ for entries of the form $(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i)})$. If it exists, $\mathcal{B}$ returns $h_2^{(i)}$ to $\mathcal{A}$, else go to next step.

2. $\mathcal{B}$ empties the list $\mathcal{L}_T$.

3. $\mathcal{B}$ then searches the list $\mathcal{L}_{H_1}$ for entries of the form $(m^{(i_n)}, r^{(i_n)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$. If such an entry exists, go to the next step, else $\mathcal{B}$ chooses a string $h_2^{(i)}$ uniformly at random from $\{0,1\}^{l_2+l_3}$ and returns $h_2^{(i)}$ to the adversary $\mathcal{A}$. $\mathcal{B}$ then adds the entry of the form $(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i)})$ into the list $\mathcal{L}_{H_2}$.

4. $\mathcal{B}$ then searches the list $\mathcal{L}_{Dec}^{In}$ for entries of the form $(c^{(i_q)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec}, ID_{Sen^{(i)}})$. If such an entry exists, go to the next step, else $\mathcal{B}$ chooses a string $h_2^{(i)}$ uniformly at random from $\{0,1\}^{l_2+l_3}$ and returns $h_2^{(i)}$ to the adversary $\mathcal{A}$. $\mathcal{B}$ then adds the entry of the form $(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i)})$ into the list $\mathcal{L}_{H_2}$.

5. For each entries of the form $(m^{(i_n)}, r^{(i_n)}, r'^{(i)}, ID_{Rec}, h_1^{(i)})$ and $(c^{(i_q)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Rec}, ID_{Sen^{(i)}})$, $\mathcal{B}$ computes $h_2^{(i_{n_q})} = m^{(i_n)}||r^{(i_n)} \oplus c^{(i_q)}$, then checks $\text{VER}(h_1^{(i_n)}, \sigma^{(i_n)}, PK_{Sen^{(i_n)}}^S) \stackrel{?}{=}$ true and $\text{ENC}_{r^{(i_n)}}(r'^{(i)}, PK_{Rec}^E) \stackrel{?}{=} c'^{(i)}$. If both satisfy, then $\mathcal{B}$ adds $h_2^{(i_{n_q})}$ into the list $\mathcal{L}_T$.

6. $\mathcal{B}$ finally chooses a string $h_2^{(i)}$ uniformly at random from $\{0,1\}^{l_2}$ such that $h_2^{(i)} \notin \mathcal{L}_T$. $\mathcal{B}$ then returns $h_2^{(i)}$ to the adversary $\mathcal{A}$ and adds the entry of the form $(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i)})$ into the list $\mathcal{L}_{H_2}$.

Claim 3: The way $\mathcal{B}$ responds to the H_2 query made by $\mathcal{A}$, it is ensured that the string, $h_2^{(i)}$, returned by $\mathcal{B}$ looks like a random string from $\mathcal{A}$'s point of view. It is also ensured that previously declared invalid ciphertexts remain invalid. Moreover, consistency of previously queried ciphertext is also maintained.

It is easy to prove the Claim 3. Proof goes along in a similar way as in Claim 2. Now we show how $\mathcal{B}$ responds to the i^{th} query if the i^{th} query is on *Designcryption*.

Designcryption **query:** For any query of the form $(c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(i)}}$, $\mathcal{B}$ follows the following algorithm to respond to $\mathcal{A}$.

Algorithm 3

1. If $\text{VER}(h_1^{(i)}, \sigma^{(i)}, PK^S_{Sen^{(i)}}) = \text{false}$, return $\perp$, else go to next step.
2. $\mathcal{B}$ searches the list $\mathcal{L}_{H_2}$ for entries of the form $(r'^{(i_n)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i_n)})$. If such an entry exists, go to next step, else return $\perp$.
3. For each entry of the form $(r'^{(i_n)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen^{(i)}}, h_2^{(i_n)})$, $\mathcal{B}$ searches in the list $\mathcal{L}_{H_1}$ for entries of the form $(m^{(i_{n_t})}, r^{(i_{n_t})}, r'^{(i_n)}, ID_{Rec}, h_1^{(i)})$. If such an entry exists, go to next step, else return $\perp$.
4. $\mathcal{B}$ empties the list $\mathcal{L}_T$
5. For each entry of the form $(m^{(i_{n_t})}, r^{(i_{n_t})}, r'^{(i_n)}, ID_{Rec}, h_1^{(i)})$, $\mathcal{B}$ checks $\text{ENC}_{r^{(i_{n_t})}}(r'^{(i_n)}, PK^E_{Rec}) \stackrel{?}{=} c'^{(i)}$ and $m^{(i_{n_t})} || r^{(i_{n_t})} \stackrel{?}{=} c^{(i)} \oplus h_2^{(i_n)}$. If both conditions hold, add $m^{(i_{n_t})}$ into $\mathcal{L}_T$.
6. If $\mathcal{L}_T$ is empty, return $\perp$, else choose one $m^{(i_{n_t})} \in \mathcal{L}_T$ and return $m^{(i_{n_t})}$ to $\mathcal{A}$.

Claim 4: The way $\mathcal{B}$ responds to $\mathcal{A}$, it is ensured that if the queried ciphertext is a valid ciphertext, $\mathcal{B}$ returns a unique message. Moreover, when $\perp$ is returned, the queried ciphertext is either invalid or $\mathcal{A}$ cannot prove the validity of the ciphertext anyhow.

It is easy to prove Claim 4. Once this stage is over, $\mathcal{A}$ enters into the challenge phase. $\mathcal{A}$ submits two equal length messages m_0 and m_1 along with receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(ch)}}$ to $\mathcal{B}$. $\mathcal{B}$ gets a challenged ciphertext, $c'^{(ch)}$, randomly generated from the encryption scheme $\mathcal{E}$ by $\mathcal{CH}$. $\mathcal{B}$ then chooses two random strings, $c^{(ch)} \in \{0,1\}^{l_2+l_3}$ and $h_1^{(ch)} \in \{0,1\}^{l_1}$ and computes $\sigma^{(ch)} = \text{SIG}(h_1^{(ch)}, SK^E_{Sen^{(ch)}}, PK^E_{Sen^{(ch)}})$. $\mathcal{B}$ then returns $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ to $\mathcal{A}$. Now, $\mathcal{A}$ again make queries on H_1, H_2 and *Designcryption* oracles. In this phase, $\mathcal{A}$ is not allowed to make a query on the challenged ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(ch)}}$ to *Designcryption* oracle. We show how $\mathcal{B}$ responds to $\mathcal{A}$ during this phase.

H_1 **query:** For a given query on $(m^{(j)}, r^{(j)}, r'^{(j)}, ID_{Rec})$, $\mathcal{B}$ follows the following algorithm to respond to $\mathcal{A}$:

Algorithm 4

1. If $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK^E_{Rec}) = c'^{(ch)}$, return $r'^{(j)}$ to $\mathcal{CH}$ and abort the game. Else, follow the Algorithm 1.

Claim 5: Algorithm 4 ensures that $\mathcal{B}$ either correctly decrypts the challenged ciphertext $c'^{(ch)}$ or if not, correctly simulates the game.

Proof of Claim 5: It is trivial to check that if $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK^E_{Rec}) = c'^{(ch)}$, then $r'^{(j)} = r'^{(ch)} = \text{DEC}(c'^{(ch)}, SK^E_{Rec}, PK^E_{Rec})$. If not, the queried tuple cannot affect the validity or invalidity of the challenged ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$. Hence, Algorithm 1 then ensures that the simulated game is correct. □

H_2 **query:** For a given query on $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}})$, $\mathcal{B}$ follows the following algorithm to respond to $\mathcal{A}$:

Algorithm 5

1. If $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}, h_2^{(j)})$ exists in $\mathcal{L}_{H_2}$, $\mathcal{B}$ returns $h_2^{(j)}$ to $\mathcal{A}$, else goes to next step.
2. If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) = (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, $\mathcal{B}$ chooses a string, $h_2^{(j)}$, uniformly at random from $\{0,1\}^{l_2+l_3}$ and puts the entry of the form $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}, h_2^{(j)})$ into the list $\mathcal{L}_{H_2}$, else follows Algorithm 2.

Claim 6: If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) = (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, then $\mathcal{B}$ correctly simulates the game if $r'^{(j)} \neq \text{DEC}(c'^{(ch)}, SK^E_{Rec}, PK^E_{Rec})$. Moreover, $\mathcal{B}$ can correctly simulate the game with probability $\frac{2}{2^{(l_2+l_3)}}$ if $\text{DEC}(c'^{(ch)}, SK^E_{Rec}, PK^E_{Rec}) = r'^{(j)}$.

Proof of Claim 6: If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) = (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, query on $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}})$ to H_2 can affect the validity of the ciphertext of the form $(c, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ only where c may or may not be equal to $c^{(ch)}$. Designcryption of the ciphertext $(c, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ requires H_2 query on $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}})$ where $\text{DEC}(c'^{(ch)}, SK^E_{Rec}, PK^E_{Rec}) = r'^{(j)}$. If $r'^{(j)}$ is not the decryption of $c'^{(ch)}$, any returned value will not affect the validity of the ciphertext of any form.

If $\text{DEC}(c'^{(ch)}, SK^E_{Rec}, PK^E_{Rec}) = r'^{(j)}$, then for any previously queried ciphertext of the form $(c, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ where $c \neq c^{(ch)}$, any returned value for H_2 query on $(r'^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}})$ will not affect the invalidity of the ciphertext unless there is a query on H_1 of the form $(m^{(j)}, r^{(j)}, r'^{(j)}, ID_{Rec})$, where $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK^E_{Rec}) = c'^{(j)} = c'^{(ch)}$ (Claim 1). If there were H_1 query on $(m^{(j)}, r^{(j)}, r'^{(j)}, ID_{Rec})$, $\mathcal{B}$ would have returned $r'^{(j)}$ earlier to $\mathcal{CH}$. If $c = c^{(ch)}$, then there could only be two such values, namely $h_2^{(j)} = m_0||r^{(j)} \oplus c^{(ch)}$ or $h_2^{(j)} = m_1||r^{(j)} \oplus c^{(ch)}$, where $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK^E_{Rec}) = c'^{(j)} = c'^{(ch)}$ for which the challenged ciphertext can remain a valid ciphertext. In such a case, the probability that $\mathcal{B}$ correctly simulates the game is $\frac{2}{2^{(l_2+l_3)}}$. □

It can be easily seen that if $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) \neq (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, Algorithm 2 then correctly simulates the game. Now, we show how $\mathcal{B}$ responds if the i^{th} query is a *Designcryption* query.

Designcryption **query:** For a given query on $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$ for receiver's identity ID_{Rec} and sender's identity $ID_{Sen^{(j)}}$, $\mathcal{B}$ follows the following algorithm to respond to $\mathcal{A}$:

Algorithm 6

1. If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) = (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, return $\perp$ to $\mathcal{A}$ and put $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Rec}, ID_{Sen^{(j)}})$ into the list $\mathcal{L}_{Dec}^{In}$, else follow Algorithm 3.

Claim 7: Algorithm 6 ensures that $\mathcal{B}$ correctly simulates the game.

Proof of Claim 7: We prove it by contradiction. Let $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) = (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$ and $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$, where $c^{(j)} \neq c^{(ch)}$, be a valid ciphertext. It can be easily verified from the designcryption algorithm that both $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$ and $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ cannot yield the same message after designcryption. Let the designcryption of the ciphertext $(c^{(j)}, c'^{(j)}, h_1^{(j)}, \sigma^{(j)})$ be $m^{(j)}$. Then $\mathcal{A}$ must have made H_1 query earlier over $(m^{(j)}, r^{(j)}, r'^{(j)})$, where $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK_{Rec}^E) = c'^{(j)} = c'^{(ch)}$ (Claim 1). In such a case, $\mathcal{B}$ would have returned $r'^{(j)}$ earlier and aborted the game.

If $(c'^{(j)}, h_1^{(j)}, \sigma^{(j)}, ID_{Sen^{(j)}}) \neq (c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, then Algorithm 3 ensures the correct simulation of the game. □

Now, $\mathcal{A}$ guesses a bit $\hat{b}$. $\mathcal{B}$ ignores the bit and searches the list $\mathcal{L}_{H_1}$ for the entry of the form $(m^{(j)}, r^{(j)}, r'^{(j)}, ID_{Rec}, h_1^{(j)})$ such that $\text{ENC}_{r^{(j)}}(r'^{(j)}, PK_{Rec}^E) = c'^{(ch)}$. If such an entry exists, $\mathcal{B}$ returns the value $r'^{(j)}$. If not, $\mathcal{B}$ then chooses randomly a value $r'^{(j)}$ from entry of the form $(r'^{(j)}, c^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$ stored in the list $\mathcal{L}_{H_2}$.

Claim 8: Let $\text{DEC}(c'^{(ch)}, SK_{Rec}^E, PK_{Rec}^E) = r'^{(ch)}$ and $\text{ENC}_{r^{(ch)}}(r'^{(ch)}, PK_{Rec}^E) = c'^{(ch)}$, then from $\mathcal{A}$'s point of view, challenge ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ can yield any of m_0 or m_1 upon designcryption unless there is an H_1 query on either $(m_0, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or $(m_1, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or an H_2 query on $(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$.

Proof of Claim 8: Suppose $\mathcal{A}$ has not made H_1 query on either $(m_0, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or $(m_1, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or H_2 query on $(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$. Then,

1. If $H_1(m_0, r^{(ch)}, r'^{(ch)}, ID_{Rec}) = h_1^{(ch)}$ and $H_2(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}}) = m_0 || r^{(ch)} \oplus c^{(ch)}$, then the challenge ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ yields m_0 upon designcryption.
2. If $H_1(m_1, r^{(ch)}, r'^{(ch)}, ID_{Rec}) = h_1^{(ch)}$ and $H_2(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}}) = m_1 || r^{(ch)} \oplus c^{(ch)}$, then the challenge ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ yields m_1 upon designcryption.

Now, we prove that $\mathcal{B}$ guesses the decryption of $c'^{(ch)}$ with probability at least $\frac{2\mathcal{A}dv(\mathcal{A})}{q_{H_2}}$, where q_{H_2} is the number of H_2 oracle queries. $\square$

From the simulation of the game and Claim 8, it is evident that unless $\mathcal{A}$ makes an H_1 query over $(m_b, r^{(ch)}, r'^{(ch)}, ID_{Rec})$, where $b \in \{0,1\}$ and $\text{ENC}_{r^{(ch)}}(r'^{(ch)}, PK^E_{Rec}) = c^{(ch)}$, or an H_2 query over $(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$, where $\text{DEC}(c'^{(ch)}, PK^E_{Rec}, SK^E_{Rec}) = r'^{(ch)}$, the challenged ciphertext $(c^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)})$ appears to be a random ciphertext from $\mathcal{A}$'s point of view. Hence $\mathcal{B}$ finds the decryption of $c'^{(ch)}$ in either $\mathcal{L}_{H_1}$ or $\mathcal{L}_{H_2}$. If it exists in $\mathcal{L}_{H_1}$, $\mathcal{B}$ finds the correct decryption of $c'^{(ch)}$ with probability 1. Else, if it exists in the list $\mathcal{L}_{H_2}$, $\mathcal{B}$ finds the correct decryption with probability at least $\frac{1}{q_{H_2}}$ as at least one of the entry in $\mathcal{L}_{H_2}$ will yield the correct decryption of $c'^{(ch)}$.

Now, let

- E_R be the event that $\mathcal{A}$ guesses the bit randomly, i.e., without making any H_1 query on either $(m_0, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or $(m_1, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or H_2 query on $(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$. In this case, $\Pr[\mathcal{A} \text{ wins}] = \frac{1}{2}$
- E_H be the event that $\mathcal{A}$ guesses the bit after making any H_1 query on either $(m_0, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or $(m_1, r^{(ch)}, r'^{(ch)}, ID_{Rec})$ or H_2 query on $(r'^{(ch)}, c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$.

Then,
$\Pr[\mathcal{A} \text{ wins}] = \Pr[\mathcal{A} \text{ wins}|E_R] \times \Pr[E_R] + \Pr[\mathcal{A} \text{ wins}|E_H] \times \Pr[E_H]$
$= \frac{1}{2} \times \Pr[E_R] + \Pr[\mathcal{A} \text{ wins}|E_H] \times \Pr[E_H]$
$= \frac{1}{2} \times (1 - \Pr[E_H]) + \Pr[\mathcal{A} \text{ wins}|E_H] \times \Pr[E_H]$ (since $\Pr[E_R] + \Pr[E_H] = 1$)

Now,
$\mathcal{A}dv(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}| = |\frac{1}{2} \times (1 - \Pr[E_H]) + \Pr[\mathcal{A} \text{ wins}|E_H] \times \Pr[E_H] - \frac{1}{2}|$
$= |\frac{\Pr[E_H]}{2}(2\Pr[\mathcal{A} \text{ wins}|E_H] - 1)| = |\frac{\Pr[E_H]}{2}||(2\Pr[\mathcal{A} \text{ wins}|E_H] - 1)|$

Hence,

$$2\mathcal{A}dv(\mathcal{A}) = \Pr[E_H]|(2\Pr[\mathcal{A} \text{ wins}|E_H] - 1)| \tag{5.1}$$

Now since, $0 \leq \Pr[\mathcal{A} \text{ wins}|E_H] \leq 1$

$$0 \leq |(2\Pr[\mathcal{A} \text{ wins}|E_H] - 1)| \leq 1 \tag{5.2}$$

Combining equations 5.1 and 5.2, we get

$$2\mathcal{A}dv(\mathcal{A}) \leq \Pr[E_H] \tag{5.3}$$

Now,
$\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{B} \text{ wins}|E_R] \times \Pr[E_R] + \Pr[\mathcal{B} \text{ wins}|E_H] \times \Pr[E_H] \geq \Pr[\mathcal{B} \text{ wins}|E_H] \times \Pr[E_H]$

From equation 5.3, we get

$$\Pr[\mathcal{B} \text{ wins}] \geq 2\Pr[\mathcal{B} \text{ wins}|E_H]\mathcal{A}dv(\mathcal{A}) \tag{5.4}$$

If E_H occurs, then $\mathcal{B}$ can find the decryption of $c'^{(ch)}$ in either $\mathcal{L}_{H_1}$ with probability 1 or in $\mathcal{L}_{H_2}$ with probability $\frac{1}{q_{H_2}}$ provided $q_{H_2} \neq 0$. Hence, in this case, probability that $\mathcal{B}$ wins with probability at least $\frac{1}{q_{H_2}}$. If $q_{H_2} = 0$, then $\mathcal{B}$ finds the decryption of $c'^{(ch)}$ in $\mathcal{L}_{H_1}$ with probability 1. In this case, $\mathcal{B}$ wins with probability 1. Hence, putting these value in equation 5.4, we get

$$\Pr[\mathcal{B} \text{ wins}] = \mathcal{A}dv(\mathcal{B}) \geq \begin{cases} 2\mathcal{A}dv(\mathcal{A}) & \text{if } q_{H_2} = 0, \\ \dfrac{2\mathcal{A}dv(\mathcal{A})}{q_{H_2}} & \text{otherwise} \end{cases}$$

♣

5.3.2 Unforgeability

Theorem 11 *If there exists an dM-EUF-iCMA adversary $\mathcal{A}$ which is able to produce a forged ciphertext during the game with a non-negligible advantage, then there exists an algorithm $\mathcal{B}$ which can forge the signature scheme $\mathcal{S}$ in the EUF-CMA game or finds a collision on hash function H_1 with a non-negligible probability. Formally,*

$$\mathcal{A}dv(\mathcal{A}) \leq Pr[\textit{forging } \mathcal{S}] + Pr[\textit{collision on } H_1]$$

Proof : We shall show how to construct an EUF-CMA adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain a non-negligible advantage against $\mathcal{S}$.

Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup algorithm of Scheme1. For fixed sender, Sen, suppose algorithm $\mathcal{B}$ receives the public key, PK^S_{Sen}, of the signature scheme $\mathcal{S}$ from $\mathcal{CH}$. As the proof is in the insider security model, we further assume that encryption scheme(s) is(are) known to $\mathcal{A}$ and public key-secret key of encryption scheme(s) for all user(s) is(are) also known to $\mathcal{A}$. $\mathcal{B}$ then sends PK^S_{Sen} to the adversary $\mathcal{A}$.

$\mathcal{B}$ keeps a list $\mathcal{L}_{Sig}$ to maintain the list of queries made by $\mathcal{A}$ to the signcryption oracle. Now, we explain how signcryption queries made by $\mathcal{A}$ are treated by $\mathcal{B}$.

Signcryption **query:** For a given i^{th} query on message $m^{(i)}$ for receiver's identity $ID_{Rec^{(i)}}$ and sender's identity ID_{Sen}, $\mathcal{B}$ follows the following algorithm to respond to $\mathcal{A}$.

Algorithm 7

1. Choose random strings $r^{(i)}, r'^{(i)} \in \mathcal{R}$.
2. Compute $H_1(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec^{(i)}})$, let its output be $h_1^{(i)}$.

3. Give $h_1^{(i)}$ to the challenger $\mathcal{CH}$ which then runs $\text{SIG}(h_1^{(i)}, SK_{Sen}^S, PK_{Sen}^S)$. Let its output be $\sigma^{(i)}$. $\mathcal{CH}$ then returns $\sigma^{(i)}$ to $\mathcal{B}$.

4. Compute $\text{ENC}(r'^{(i)}, PK_{Rec}^E)$. Let its output be $c'^{(i)}$.

5. Compute $H_2(r'^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}, ID_{Sen})$. Let its output be $h_2^{(i)}$.

6. Compute $c^{(i)} = m^{(i)}||r^{(i)} \oplus h_2^{(i)}$.

7. Add $m^{(i)}$ into the list $\mathcal{L}_{Sig}$.

8. Return $(c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)})$ to $\mathcal{A}$.

Once this stage is over, $\mathcal{A}$ submits the forged ciphertext $(c^{(fg)}, c'^{(fg)}, h_1^{(fg)}, \sigma^{(fg)})$ for receiver's identity $ID_{Rec^{(fg)}}$ and sender's identity ID_{Sen} to $\mathcal{B}$. Now, the following cases arise:

1. For any $c^{(i)}$, $c'^{(i)}$ and $\sigma^{(i)}$, $(c^{(i)}, c'^{(i)}, h_1^{(fg)}, \sigma^{(i)})$ has not been returned by $\mathcal{B}$ to $\mathcal{A}$. In this case, $\mathcal{B}$ will return $(h^{(fg)}, \sigma^{(fg)})$ to $\mathcal{S}$.

2. $(c^{(i)}, c'^{(i)}, h_1^{(fg)}, \sigma^{(i)})$ has been returned by $\mathcal{B}$ to $\mathcal{A}$ for some $c^{(i)}$, $c'^{(i)}$ and $\sigma^{(i)}$ ($c^{(i)}$, $c'^{(i)}$ or $\sigma^{(i)}$ may be equal to $c^{(fg)}$, $c'^{(fg)}$ or $\sigma^{(fg)}$). Let
   $m^{(fg)} = \text{DSC}((c^{(fg)}, c'^{(fg)}, h_1^{(fg)}, \sigma^{(fg)}), SK_{Rec^{(fg)}}^E, PK_{Rec^{(fg)}}^E, PK_{Sen}^S)$
   $m^{(i)} = \text{DSC}((c^{(i)}, c'^{(i)}, h_1^{(i)}, \sigma^{(i)}), SK_{Rec^{(i)}}^E, PK_{Rec^{(i)}}^E, PK_{Sen}^S)$
   $r'^{(fg)} = \text{DEC}(c'^{(fg)}, SK_{Rec^{(fg)}}^E, PK_{Rec^{(fg)}}^E)$
   $r'^{(i)} = \text{DEC}(c'^{(i)}, SK_{Rec^{(i)}}^E, PK_{Rec^{(i)}}^E)$
   and $\text{ENC}_{r^{(fg)}}(r'^{(fg)}, PK_{Rec^{(fg)}}^E) = c'^{(fg)}$
   $\text{ENC}_{r^{(i)}}(r'^{(i)}, PK_{Rec^{(i)}}^E) = c'^{(i)}$

   Now again, the following two cases arise:

   - $m^{(fg)} \neq m^{(i)}$. In this case, $H_1(m^{(fg)}, r^{(fg)}, r'^{(fg)}, ID_{Rec^{(fg)}}) = h_1^{(fg)} = H_1(m^{(i)}, r^{(i)}, r'^{(i)}, ID_{Rec^{(i)}})$. A collision is obtained on H_1.
   - $m^{(fg)} = m^{(i)}$ implies $m^{(fg)} \in \mathcal{L}_{Sig}$. Such a case is not allowed as per the definition of dM-EUF-iCMA.

It is evident from the above analysis that whenever $\mathcal{A}$ submits a forged ciphertext, $\mathcal{B}$ either forges a message on $\mathcal{S}$ or finds a collision on H_1. Hence,

$$\mathcal{A}dv(\mathcal{A}) \leq \Pr[\text{forging } \mathcal{S}] + \Pr[\text{collision on } H_1]$$

♣

5.4 Proposed Scheme : Scheme 2

Let $\mathcal{E} = (\mathrm{KG}^E, \mathrm{ENC}, \mathrm{DEC})$ and $\mathcal{S} = (\mathrm{KG}^S, \mathrm{SIG}, \mathrm{VER})$ be encryption and signature schemes respectively. Let l_1 be the bit-length of the signing message used in the algorithm SIG of $\mathcal{S}$. Let l_2 be the bit-length of any message m from the message space $\mathbf{M}$. Moreover, let $\mathbf{R}$ be the random space .

The construction of Scheme 2 is described in Table 5.2.

Scheme 2	
Setup(1^λ) • Choose two cryptographically secure hash functions $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $H_2 : \{0,1\}^* \to \{0,1\}^{l_2}$ where l_1 = length of the signing message used in the signature scheme. l_2 = length of the message m. $Params \leftarrow (H_1, H_2, 1^\lambda)$	**SC**$(m, PK_{Rec}, PK_{Sen}, SK_{Sen}, Params)$ • $r' \leftarrow \mathbf{R}$ • $h_1 = H_1(m, r', ID_{Rec})$ • $c' \leftarrow \mathrm{ENC}(r', PK^E_{Rec})$ • $\sigma \leftarrow \mathrm{SIG}(h_1, SK^S_{Sen}, PK^S_{Sen})$ • $h_2 = H_2(r', c', h_1, \sigma, ID_{Sen})$ • $c = h_2 \oplus m$ Ciphertext $\mathcal{C} \equiv (c, c', h_1, \sigma)$
KG$_{Rec}(Params)$ **&** **KG**$_{Sen}(Params)$ • $(PK^E, SK^E) \leftarrow \mathrm{KG}^E(1^\lambda)$ • $(PK^S, SK^S) \leftarrow \mathrm{KG}^S(1^\lambda)$ • $PK_{Rec} = (PK^E_{Rec}, PK^S_{Rec})$ • $SK_{Rec} = (SK^E_{Rec}, SK^S_{Rec})$ • $PK_{Sen} = (PK^E_{Sen}, PK^S_{Sen})$ • $SK_{Sen} = (SK^E_{Sen}, SK^S_{Sen})$	**DSC**$(\mathcal{C} \equiv (c, c', h_1, \sigma), SK_{Rec}, PK_{Rec}, PK_{Sen}, Params)$ • $r' \leftarrow \mathrm{DEC}(c', SK^E_{Rec}, PK^E_{Rec})$ • $x \leftarrow \mathrm{VER}(h_1, \sigma, PK^S_{Sen})$. r' is $\perp$ or x is false, return $\perp$; else go to the next step. • $h_2 = H_2(r', c', h_1, \sigma, ID_{Sen})$ • $m = h_2 \oplus c$ • if $h_1 = H_1(m, r', ID_{Rec})$, return m, else $\perp$.

Table 5.2: Construction of Scheme 2

5.5 Security of Scheme 2

5.5.1 Confidentiality

In this section, we prove the security of *Scheme 2*. In this scheme, the proof of security requires the encryption scheme to be OWE-PCA secure. In *Scheme 1*, during designcryption phase, there is a step of checking the integrity of the ciphertext c' by running the encryption algorithm on r' using r as a random string, but it is not possible in *Scheme 2*. Such gap in the security proof of *Scheme 2* can be filled if a plaintext-checking oracle is provided.

Theorem 12 *In the random oracle model, if there exists a dM-IND-iCCA adversary $\mathcal{A}$ which can distinguish ciphertexts during the relevant game with a non-negligible advantage, then there exists an algorithm $\mathcal{B}$ which can break the OWE-PCA security of the parent encryption scheme $\mathcal{E}$ with a non-negligible advantage. Formally,*

$$Pr[\mathcal{B}\ wins] = \mathcal{A}dv(\mathcal{B}) \geq 2\mathcal{A}dv(\mathcal{A})$$

Proof : The proof goes in a similar way as it has been done for *Scheme 1* in the subsection of confidentiality (see subsection 5.3.1). In this proof, $\mathcal{B}$ interacts with both $\mathcal{A}$ and $\mathcal{CH}$. The differences between this proof and the proof done in subsection 5.3.1 are as follows:

- Instead of checking $ENC_r(r', PK^E_{Rec}) \stackrel{?}{=} c'$, $\mathcal{B}$, here, gives (r', c') to *plaintext checking oracle*.
  - If the returned value is 1 (r' is the decryption of c'), $\mathcal{B}$ behaves in a similar manner as if $ENC_r(r', PK^E_{Rec})$=c' in the proof of *Scheme 1*.
  - If the returned value is 0 (r' is not the decryption of c'), $\mathcal{B}$ behaves in a similar manner as if $ENC_r(r', PK^E_{Rec}) \neq c'$ in the proof of *Scheme 1*.
- After the challenge phase, whenever $\mathcal{A}$ submits $(r', c'^{(ch)}, h_1^{(ch)}, \sigma^{(ch)}, ID_{Sen^{(ch)}})$ to $\mathcal{B}$ while querying on H_2, $\mathcal{B}$ submits $(r', c'^{(ch)})$ to *plaintext checking oracle*. If the returned value is 1, $\mathcal{B}$ aborts the game and returns r' to $\mathcal{CH}$ else follows the algorithm specified in the proof of *Scheme 1*.

5.5.2 Unforgeability

Theorem 13 *If there exists an dM-EUF-iCMA adversary $\mathcal{A}$ which is able to produce a forged ciphertext during the game with a non-negligible advantage, then there exists an algorithm $\mathcal{B}$ which can forge the signature scheme $\mathcal{S}$ in the EUF-CMA game or finds a collision on hash function H_1 with a non-negligible probability. Formally,*

$$\mathcal{A}dv(\mathcal{A}) \leq Pr[forging\ \mathcal{S}] + Pr[collision\ on\ H_1]$$

Proof : This proof also goes in a similar way as it has been done for *Scheme 1* in the subsection of unforgeability (see subsection 5.3.2).

Chapter 6

Ring Signature with Designated Verifier for Signer-Identity

A ring signature scheme enables a user to sign a message such that a verifier would identify a "ring" of possible signers (of which the user is a member), but unable to determine the identity of the actual signer. This is true even if the verifier is a ring member. In this chapter, we examine a situation in that the signer Alice can produce a ring signature for Bob (may or may not be a ring member) such that,

- any body can verify the ring signature, and
- *only* Bob can verify the identity of Alice, provided she outputs a trapdoor information.
- Alice can *publicly*[1] output the trapdoor, i.e., this additional information does not reveal her identity to the other ring members.

We name this model as "*ring signature with designated verifier for signer-identity*". The existing ring signature variants do not capture the functionality offered by this new model. We formulate the security requirements for this new ring signature variant and finally propose a secure ring signature with designated verifier for signer-identity (RS-DVSI) scheme.

6.1 Introduction

Ring signatures enable a user to sign a message so that a "ring" of possible signers (of which the user is a member) is identified, without revealing exactly which member of that ring actually generated the signature. This notion was first formalized by Rivest, Shamir and Tauman [82], and ring signatures, along with several close variants, have been studied extensively since then [1, 19, 72, 93, 17, 51, 66, 37, 67, 92, 61, 28, 13, 41]. A ring signature scheme can be considered as a simplified group signature scheme [24] that have only users and no *managers*. Both group signatures and ring signatures are *signer-ambiguous*, but in a ring signature scheme there are no prearranged group of users, there are no procedures for

[1]She can sign the trapdoor information with the same ring of users. Thus Bob would know that both pieces of information, the earlier message and the trapdoor, has been originated from the same set of users.

setting, changing, or deleting groups of users, there is no way to distribute specialized keys, and there is no way to revoke the anonymity of the actual signer.

Ring signatures lend themselves to a variety of applications. The original motivation of ring signatures was to allow secrets to be leaked anonymously. For example, ring signature scheme will allow a high-ranking government official to sign certain information with respect to the ring of similar officials such that the information can later be verified as coming from one of the officials without exposing the actual signer (Fig 6.1 describes the same for handwritten documents).

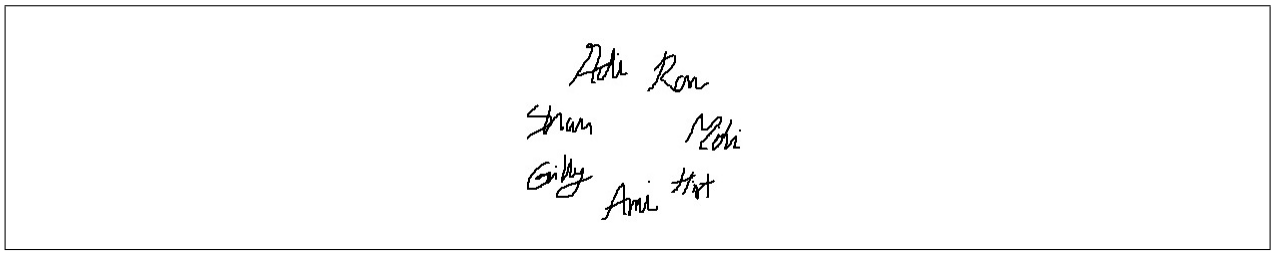


Figure 6.1: Reveals no information about the leader (the one who has signed first !!).

Ring signatures has also application to designated-verifier signatures [55]. A designated verifier signature scheme is a signature scheme in which signatures can only be verified by a *designated verifier* chosen by the signer. This can be easily achieved by using a ring signature scheme with sender and receiver as the only ring members: the sender sign the message with respect to the ring containing the sender and the receiver; the receiver is then assured that the message originated from the sender but cannot prove this to any third party, since the receiver could have produced this message-ring signature pair by itself.

6.1.1 Motivation and Our Contribution

Ring signatures are *signer-ambiguous*. However, for certain applications, the signer may like to claim his authorship on the anonymized ring signature at a later stage. A necessary requirement for such a realization is an additional input from the signer at a later stage (the ring signature alone is signer ambiguous). The stage of verification for signer's identity can be one of the following:

1. At a later stage, given an additional input from the signer, the identity of the signer can be verified by anybody[2].

2. At a later stage, given an additional input from the signer, the identity of the signer can *only* be verified by a *designated verifier*. The designated verifier can be a ring member or a third party. This case further yields two important possibilities:

[2]For example, *traceable ring signature* is a ring signature scheme where each message is signed not only with respect to a list of ring members, but also with respect to an *issue*. If a user signs any two messages with respect to the same list of ring members and the same issue label, the signer's identity is revealed by an efficient public procedure

(a) The additional input could reveal the identities of the signer and the designated verifier to the other ring members.

(b) The additional input still guarantees the privacy of their identities (signer and designated verifier) to the other ring members.

The subtle differences among the above cases become prominent for the appropriate applications. The cases 1 and 2(a) were addressed in the literature. To the best of our knowledge, the functionality of case 2(b) was not discussed earlier. We observe that this functionality cannot be realized by the existing variants of ring signatures. We first describe a practical problem to further illustrate the usefulness of 2(b).

Let us suppose, Bob is a layer-2 governing body member (consisting of say n members) of a company and the layer-1 body features some powerful members (say r, where $r < n$) among the members of layer-2 body. Alice happens to be a member of the layer-1 body. A certain unethical decision has been taken in a layer-1 body meeting and Bob would like to get this information. We further assume that Alice is aware of this fact and wish to leak the information to Bob. We assume, no secure channels exists among members; therefore the communication should take place on authenticated public channels (viz. bulletin boards). Alice needs to carry out this task in a manner such that Bob gets convinced about the information being originated from a layer-1 committee member. It is required that identities of Bob and Alice should not be leaked to any third party. Further, without any help from Alice, the signature alone should not give any information to Bob about the identity of Alice. Bob can later verify the identity of Alice (for reward) with an additional input from Alice (this additional information can be further ring-signed and sent over public channel).

We now discuss several available variants of ring signatures and argue about their inability to address the above scenario.

- Designated Verifier Signature Scheme [55]: In this primitive, the signature reveals the identity of the designated verifier.

- Step-Out Ring Signature [58]: This serves as an intermediate between ring and group signature scheme. In a step-out ring signature scheme, the anonymity status of the signer can be changed by two procedures - Confession and Step-Out. In Confession algorithm, the signer can reveal his/her identity by producing another signature that can be verified publicly. In Step-Out procedure, a member of the ring that is not the signer can prove that he/she is not the signer. In this procedure, a non-signer produces a step-out record which is a signature on some message $\bar{m}$ = "*I have not signed* m". Thus, it is clear that the step-out ring signature will not work.

- Strongly Designated Verifier Ring Signature [60]: In such a scheme the signer designate the verifier at the time of signature generation. None except the designated verifier (DV) can verify the signature but the designated verifier can neither break the anonymity of the scheme nor can convince the third party about the actual signer of the message.

- Universal Designated Verifier Ring Signature [62]: This scheme not allows members of a group to sign messages on behalf of the group without revealing their identities, but also allows any holder of the signature to designate the signature to any designated verifier. In this manner, either it preserves the anonymity of the signer (actual signer can not reveal his/her identity) or if it can be revealed, it can be done by designated verifier once the signature is made public.

- Controllable Ring Signature [44]: This is a ring signature scheme with additional properties - (a) Anonymous identification: the real signer can anonymously prove his authorship to the verifier using an anonymous identification protocol. (b) Linkable signature: newly generated anonymous signature should be linked to a previously generated anonymous signature so that it can be verified publicly that both the signatures have been generated by the same user. (c) the ring signature can be converted to an ordinary signature after the revelation of secret information by the real signer. Thus using controllable ring signature Alice can leak some secret information anonymously and publicly, but she can not prove her identity as a signer to Bob at a later stage as

  - Anonymous identification allows her to prove her authorship anonymously. Therefore, Bob can not be convinced at a later stage that Alice is the real signer.
  - Convertibility allows her to reveal her identity publicly by converting the ring signature into an ordinary signature.

RS-DVSI has three stages - (i) The generation of ring signature σ by the signer u_s designated for the verifier u_d on some message m for some chosen ring of members $\mathcal{R}$ (ii) the trapdoor sv_{u_s} is made public by the actual signer u_s and (iii) with the help of trapdoor, the designated verifier u_d knows the identity of the actual signer u_d. Note that, although anybody can verify the ring signature σ on message m and $\mathcal{R}$, anonymity of the actual signer is still preserved until the trapdoor is revealed. Hence, the actual signer, upon his/her discretion only, can reveal his/her identity. In the case of "Strongly Designated Verifier Ring Signature" or "Universal Designated Verifier Ring Signature", there is no step (ii). Step-Out Ring Signature has three stages but it leaks the signer's identity publicly. Similarly, controllable ring signature also has three stages but it either maintains the anonymity of the signer or reveals the identity of the signer publicly. In nutshell, to the best of our knowledge, there does not exist any scheme in the literature which serves the same purpose as RS-DVSI does.

In this chapter we formalize a variant notion of ring signature that will address the case 2(b). We call this new notion "**Ring Signature with Designated Verifier for Signer-Identity (RS-DVSI)**". We then formulate the appropriate security requirements for a RS-DVSI scheme. Finally we propose a secure scheme.

6.2 Ring Signature with Designated Verifier for Signer-Identity (RS-DVSI)

We begin by presenting the definition of a ring signature scheme with designated verifier for signer-identity. We refer to an ordered list $\mathcal{R} = (PK_{u_1}, \ldots, PK_{u_n})$ of public keys as a

ring, and let $\mathcal{R}[i] = PK_{u_i}$. Let u_i be the corresponding user whose public key is PK_{u_i}: for example we denote u_s and u_d as the signer and the designated verifier respectively. A ring signature scheme with designated verifier for signer-identity consists of five algorithms (Setup, KeyGen, RingSign, RingVerify, RevealTrapdoor, IdentityVerification) that, respectively, generate public parameters and master secret key, generate keys for a user, sign a message, verify the signature of a message, reveals a trapdoor information and verify the identity of the signer by the designated verifier. Formally,

- **Setup(1^λ):** The probabilistic polynomial time algorithm takes as input the security parameter 1^λ and outputs a public parameter $Params$ and a master secret key MSK. (MSK may be an empty string).

- **KeyGen($ID, MSK, Params$):** The probabilistic polynomial time *user key generation* algorithm takes as input the user's identity ID, the master secret key MSK (if non-empty) & the public parameter $Params$ and outputs a public key PK and a secret key SK.

- **RingSign($m, \mathcal{R}, SK_{u_s}, PK_{u_d}, Params$):** The probabilistic polynomial time *ring signing* algorithm takes as input a message m, the ring $\mathcal{R}$ that is a set of public keys (such that $PK_{u_s} \in \mathcal{R}$), signer's secret key SK_{u_s}, designated verifier's public key PK_{u_d}, public parameter $Params$ and to return a signature σ on m with respect to the ring $\mathcal{R}$ and the designated verifier with the identity u_d.

- **RingVerify($m, \sigma, \mathcal{R}, Params$):** The deterministic polynomial time *ring verification* algorithm takes as input a message m, a signature σ and the ring $\mathcal{R}$ to return a single bit $b \in \{0, 1\}$. If σ is a valid signature on message m for the ring $\mathcal{R}$, b is 1 else 0.

- **RevealTrapdoor($m, \sigma, \mathcal{R}, SK_{u_s}, PK_{u_d}, Params$):** The probabilistic *reveal trapdoor* algorithm takes as input a message m, a signature σ for m, the ring $\mathcal{R}$, the secret key SK_{u_s} of the signer who have produced σ, and the public key PK_{u_d} of the designated verifier to return a trapdoor string sv_{u_s}. The trapdoor string is made public to all the members of $\mathcal{R}$ by the signer.

- **IdentityVerification($m, \sigma, \mathcal{R}, PK_{u_j}, SK_{u_j}, sv_{u_s}, Params$):** The deterministic polynomial time *signer identity verification* algorithm does the exhaustive search over all the public keys in $\mathcal{R}$. For every public key from $\mathcal{R}$, it takes as input the message m, the signature σ for m, the public key-secret key pair (PK_{u_j}, SK_{u_j}) of the user, and the trapdoor value sv_{u_s} to return either u_s if σ was generated by u_s on message m for the ring $\mathcal{R}$ designated for u_j, else returns $\perp$.

6.3 Security

In this section we formally define all the security notions by means of standard security games between the challenger and the adversary. In such a game, specific capabilities of the adversary will become clear.

6.3.1 Anonymity

The anonymity condition for a regular ring signature scheme requires, informally that an adversary should not be able to tell which member of a ring generated a particular signature with probability greater than $\frac{1}{|\mathcal{R}|}$. This limited anonymity can be either *computational* or *unconditional*. In the model of ring signatures with designated verifier for signer-identity, it requires the signature to be both *signer and designated verifier-ambiguous*. We consider unconditional anonymity for our proposed scheme. We point out that our anonymity definition does not allow *adversarially-chosen* public keys in the ring $\mathcal{R}$, a stronger definition for anonymity then that has been formulated recently by Bender, Katz, and Morselli [13].

Definition 1 (Unconditional Anonymity) *The adversary with unbounded computation power is given a signature computed by a randomly-chosen signer from the ring $\mathcal{R}$, with the requirement that the adversary should be unable to guess the actual signer and the designated verifier (assuming them to be different) with probability better than $\frac{1}{2\binom{|\mathcal{R}|}{2}} + \epsilon$, where ϵ is a negligible function of the security parameter.*

6.3.2 Unforgeability

The intuitive notion of unforgeability is, as usual, that an adversary should be unable to output $(m, \sigma, \mathcal{R})$ such that $\mathsf{RingVerify}(m, \sigma, \mathcal{R}, Params) = 1$. We now define the *unforgeability against chosen-subring attacks* for a RS-DVSI scheme.

Definition 2 (Unforgeability Against Chosen-Subring Attacks (UF-CSA))

A ring signature scheme with designated verifier for signer-identity is unforgeable against chosen-subring attacks if for any PPT adversary $\mathcal{A}$ and for any polynomial $\ell(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible:

- Key pairs $\{(PK_{u_i}, SK_{u_i})\}_{i=1}^{\ell(\lambda)}$ are generated using $\mathsf{KeyGen}(ID, MSK, Params)$, and the set of public keys $\mathcal{R} \stackrel{\text{def}}{=} \{PK_{u_i}\}_{i=1}^{\ell(\lambda)}$ along with public parameters $Params$ is given to $\mathcal{A}$.
- $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{OSign}(\cdot, \cdot, \cdot, \cdot)$, where $\mathsf{OSign}(m, \mathcal{R}', u_s, u_d)$ outputs the ring signature $\sigma \leftarrow \mathsf{RingSign}(m\mathcal{R}', SK_{u_s}, PK_{u_d}, Params)$ along with the trapdoor generated by $\mathsf{RevealTrapdoor}(m, \sigma, \mathcal{R}', SK_{u_s}, PK_{u_d}, Params)$ and we require that $\mathcal{R}' \subseteq \mathcal{R}$, $PK_{u_s} \in \mathcal{R}'$ and $PK_{u_d} \in \mathcal{R}$.
- $\mathcal{A}$ outputs $(m^*, \sigma^*, \mathcal{R}^*)$ and succeeds if $\mathcal{R}^* \subseteq \mathcal{R}$, $\mathsf{RingVerify}(m^*, \sigma^*, \mathcal{R}^*, Params) = 1$, and $\mathcal{A}$ never queried $(m^*, \mathcal{R}^*, \star, \star)$ to its signing oracle.

We define the advantage of $\mathcal{A}$ in the above game as $\mathcal{A}dv(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins the game}]$.

6.3.3 Uniqueness of Signer and Designated Verifier

Definition 3

An RS-DVSI scheme is said to preserve signer's and designated identity verifier's uniqueness if for any probabilistic polynomial time adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the following game is negligible:

- $\mathcal{A}$ queries for public and secret key of different users. Then key pairs $\{(PK_{u_i}, SK_{u_i})\}_{i=1}^{p(\lambda)}$ are generated using $\mathsf{KeyGen}(ID, MSK, Params)$, and the set of public keys $\mathcal{R} \stackrel{\text{def}}{=} \{PK_{u_i}\}_{i=1}^{p(\lambda)}$, set of secret keys $\{SK_{u_i}\}_{i=1}^{p(\lambda)}$ and public parameters $Params$ are given to $\mathcal{A}$.
- $\mathcal{A}$ outputs $(m, \sigma, \mathcal{R}')$, (u_{s_0}, u_{d_0}, r_0) & (u_{s_1}, u_{d_1}, r_1) and succeeds if, $\mathsf{IdentityVerification}$ algorithm returns u_{s_0} for input $(m, \sigma, \mathcal{R}', SK_{u_{d_0}}, PK_{u_{d_0}}, r_0, Params)$ and returns u_{s_1} for input $(m, \sigma, \mathcal{R}', SK_{u_{d_1}}, PK_{u_{d_1}}, r_1, Params)$ where $\mathcal{R}' \subseteq R$, $\{PK_{u_{s_0}}, PK_{u_{s_1}}\} \subseteq \mathcal{R}'$, $\{PK_{u_{d_0}}, PK_{u_{d_1}}\} \subseteq \mathcal{R}$ and $\{u_{s_0}, u_{d_0}\} \neq \{u_{s_1}, u_{d_1}\}$.

We define the advantage of $\mathcal{A}$ in the above game as $\mathcal{A}dv(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins the game}]$.

6.3.4 Signer's Privacy

Intuitively, a secure RS-DVSI scheme should hide the signer's identity from the adversary who is given both the ring signature and the corresponding trapdoor information.

Definition 4

An RS-DVSI scheme is said to preserve signer's privacy if for any PPT adversary $\mathcal{A}$ and for any polynomial $\ell(\cdot)$, the probability that $\mathcal{A}$ wins the following game is negligible:

1. Key pairs $\{(PK_{u_i}, SK_{u_i})\}_{i=1}^{\ell(\lambda)}$ are generated using $\mathsf{KeyGen}(ID, MSK, Params)$, and the set of public keys $\mathcal{R} \stackrel{\text{def}}{=} \{PK_{u_i}\}_{i=1}^{\ell(\lambda)}$ along with public parameters $Params$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *corrupt oracle* $\mathsf{Corrupt}(\cdot)$, where $\mathsf{Corrupt}(u_i)$ outputs SK_{u_i}.
3. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{OSign}(\cdot, \cdot, \cdot, \cdot)$, where $\mathsf{OSign}(m, \mathcal{R}', u_s, u_d)$ outputs the ring signature $\sigma \leftarrow \mathsf{RingSign}(m, \mathcal{R}', SK_{u_s}, PK_{u_d}, Params)$ along with the trapdoor generated by $\mathsf{RevealTrapdoor}(m, \sigma, \mathcal{R}', SK_{u_s}, PK_{u_d}, Params)$ and we require that $\mathcal{R}' \subseteq \mathcal{R}$, $PK_{u_s} \in \mathcal{R}'$ and $PK_{u_d} \in \mathcal{R}$.
4. $\mathcal{A}$ then outputs a tuple $(m, \mathcal{R}^*, u_{s_0}, u_{s_1}, u_d)$, $\mathcal{R}^* \subseteq \mathcal{R}$ and $\{PK_{u_{s_0}}, PK_{u_{s_1}}\} \in \mathcal{R}^*$, $PK_{u_d} \in \mathcal{R}$. To this, a bit $b \in \{0, 1\}$ is chosen uniformly at random and $\mathcal{A}$ is given ring signature $\sigma = \mathsf{RingSign}(m, \mathcal{R}^*, SK_{u_{s_b}}, PK_{u_d}, Params)$ and the corresponding trapdoor value $sv = \mathsf{RevealTrapdoor}(m, \sigma, \mathcal{R}^*, SK_{u_{s_b}}, PK_{u_d}, Params)$.
5. Finally, $\mathcal{A}$ returns a bit $\hat{b} \in \{0, 1\}$. $\mathcal{A}$ wins the game if $b = \hat{b}$ and $\mathcal{A}$ did not query $\mathsf{Corrupt}(\cdot)$ on u_{s_0}, u_{s_1} and u_d.

We define the advantage of $\mathcal{A}$ in the above game as $\mathcal{A}dv(\mathcal{A}) = |\Pr[b = \hat{b}] - \frac{1}{2}|$.

6.3.5 Designated Verifier's Privacy

We define the privacy of a designated verifier in a RS-DVSI scheme similar to the Definition 4. The security game is the same as for the Signer's privacy, except the following changes in step 5-6.

5. $\mathcal{A}$ then outputs a tuple $(m, \mathcal{R}^*, u_s, u_{d_0}, u_{d_1})$, $\mathcal{R}^* \subseteq \mathcal{R}$ and $\{PK_{u_{d_0}}, PK_{u_{d_1}}\} \subseteq \mathcal{R}$, $PK_{u_s} \in \mathcal{R}^*$. To this, a bit $b \in \{0,1\}$ is chosen uniformly at random and $\mathcal{A}$ is given ring signature $\sigma = \mathsf{RingSign}(m, \mathcal{R}^*, SK_{u_s}, PK_{u_{d_b}}, Params)$ and the corresponding trapdoor value $sv = \mathsf{RevealTrapdoor}(m, \sigma, \mathcal{R}^*, SK_{u_s}, PK_{u_{d_b}}, Params)$.

6. Finally, $\mathcal{A}$ returns a bit $\hat{b} \in \{0,1\}$. $\mathcal{A}$ wins the game if $b = \hat{b}$ and $\mathcal{A}$ did not query $\mathsf{Corrupt}(\cdot)$ on u_{d_0}, u_{d_1} and u_s.

6.4 Generic Construction of RS-DVSI

Let **RS** = (Setup^{RS}, KeyGen^{RS}, RSign, RVerify) be a ring signature scheme, **IBSC** = (Setup^{IBSC}, KeyGen^{IBSC}, Signcrypt, Designcrypt) be an Identity Based Signcryption scheme, $H_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $H_2 : \{0,1\}^* \to \{0,1\}^{l_2}$ be two hash functions. Let Signcrypt algorithm of **IBSC** take a k bit message and return an l_1 bit ciphertext. The proposed **RS-DVSI** scheme is as follows:

- $\mathsf{Setup}(1^\lambda)$
  - $Params^{RS} \leftarrow \text{Setup}^{RS}(1^\lambda)$
  - $(MSK, Params^{IBSC}) \leftarrow \text{Setup}^{IBSC}(1^\lambda)$
  - $Params \leftarrow (Params^{RS}, Params^{IBSC})$
- $\mathsf{KeyGen}(ID, MSK, Params)$.
  - $(PK^{RS}, SK^{RS}) \leftarrow \text{KeyGen}^{RS}(ID, Params^{RS})$
  - $SK^{IBSC} \leftarrow \text{KeyGen}^{IBSC}(ID, MSK, Params^{IBSC})$
  - $PK = (PK^{RS}, ID)$ and $SK = (SK^{RS}, SK^{IBSC})$
- $\mathsf{RingSign}(m, \mathcal{R} = (\mathcal{R}_1, \mathcal{R}_2) = (\cup_{i=1}^n \{PK_{u_i}^{RS}\}, \cup_{i=1}^n \{ID_{u_i}\}), SK_{u_s}, PK_{u_d}, Params)$
  - $\sigma_1 = \text{RSign}(m, \mathcal{R}_1, SK_{u_s}^{RS}, Params^{RS})$
  - Let $M = m||\sigma_1||\mathcal{R} = m_1||m_2||...||m_z$ where $|m_i| = k$ for $i = 1, \ldots, z$. (if M is not a multiple of k then pad it with some string).
  - For $i = 1$ to z
    - Choose a random string r_i.
    - $\sigma_{2,i} = \text{Signcrypt}(m_i, SK_{u_s}^{IBSC}, ID_{u_d}, ID_{u_s}, Params^{IBSC}) \oplus H_1(r_i)$
  - $\sigma_3 = H_2(r_1|| \ldots ||r_z)$

– Ring-Signature $\sigma = (\sigma_1, \sigma_2 = (\sigma_{2,1}||\ldots||\sigma_{2,z}), \sigma_3)$.

- RingVerify$(m, \sigma, \mathcal{R}, Params)$
  - $x \leftarrow \text{RVerify}(m, \sigma_1, \mathcal{R}_1, Params^{RS})$
- RevealTrapdoor$(m, \sigma, \mathcal{R}, SK_{u_s}, PK_{u_d}, Params)$
  - $sv_{u_s} \leftarrow (r_1, \ldots, r_z)$
- IdentityVerification$(m, \sigma, \mathcal{R}, PK_{u_j}, SK_{u_j}, sv_{u_s}, Params)$

  For $l = 1$ to n $(ID_l \in \mathcal{R}_2)$
  - For $i = 1$ to z
    - $y_i \leftarrow \text{Designcrypt}((\sigma_{2,i} \oplus H_1(r_i)), SK_{u_j}^{IBSC}, ID_{u_j}, ID_{u_l}, Params^{IBSC})$
    - Check $(y_i \stackrel{?}{=} m_i)$
  - If true for all $i \in \{1, \ldots, z\}$ and $H_2(r_1||\ldots||r_z) = \sigma_3$ then return u_l

  Else, Return $\perp$

6.4.1 Correctness of the Scheme

Correctness of the scheme trivially follows from the correctness of the ring signature scheme **RS** and the consistency of the Identity Based Signcryption scheme **IBSC**. But to ensure that u_d only should be able to verify the identity of the signer u_s with overwhelming probability if σ is produced by u_s for u_d, one property on IBSC scheme is required, *identity collision-resistant* (see chapter 2, subsection 2.9.2), which means with negligible probability there exists two different sets of identities $\{u_{s_0}, u_{d_0}\} \neq \{u_{s_1}, u_{d_1}\}$ such that $\text{Signcrypt}(m, SK_{u_{s_0}}^{IBSC}, ID_{u_{d_0}}, ID_{u_{s_0}}, Params^{IBSC}) = \text{Signcrypt}(m, SK_{u_{s_1}}^{IBSC}, ID_{u_{d_1}}, ID_{u_{s_1}}, Params^{IBSC})$ for randomly chosen message m and all random coins used in the algorithm 'Signcrypt'. In our proposed RS-DVSI scheme, we choose **IBSC** to be *identity collision-resistant*.

6.5 Security

For the sake of simplicity of notations, throughout the security analysis, we denote by $C \leftarrow \text{Signcrypt}((m||\sigma||\mathcal{R}), SK_{u_i}^{IBSC}, ID_{u_i}, ID_{u_j}, Params^{IBSC})$ if $(m||\sigma||\mathcal{R}) = (m_1||\ldots||\ m_z)$ and $C = C_1||\ldots||C_z$ where $C_l \leftarrow \text{Signcrypt}(m_l, SK_{u_i}^{IBSC}, ID_{u_i}, ID_{u_j}, Params^{IBSC})$ for all $l \in \{1, \ldots, z\}$.

Theorem 14 *In the random oracle model, assuming that H_1 and H_2 as random functions, the proposed RS-DVSI scheme is unconditionally anonymous if the ring signature scheme RS is unconditionally anonymous.*

Proof : It is clear from the construction that in ring signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$, if r_i is chosen randomly, $H(r_i)$ also behaves as a random string (in the random oracle model), hence $\sigma_2 = \text{Signcrypt}((m||\sigma_1||\mathcal{R}), SK_{u_s}^{IBSC}, ID_{u_d}, ID_{u_s}, Params^{IBSC}) \oplus H_1(r_1)|| \ldots ||H_1(r_z)$ behaves as a random string i.e. independent of any user's identities. Similarly, $\sigma_3 = H_2(r_1|| \ldots ||r_z)$ also behaves as a random string. Hence, the anonymity of the σ then depends upon σ_1 only. Hence, if the ring signature RS is unconditionally anonymous, the whole $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ preserves unconditional anonymity.

Theorem 15 *If there exists an adversary $\mathcal{A}$ which can break the UF-CSA security of the proposed RS-DVSI scheme, then there exists an adversary $\mathcal{B}$ which can break the unforgeability of the ring signature scheme RS. Moreover, $\mathcal{A}dv(\mathcal{B}) \geq \mathcal{A}dv(\mathcal{A})$.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{RS} and KeyGen^{RS} of the ring signature scheme RS. We shall show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the unforgeability of RS. Suppose $\mathcal{B}$ receives public keys $\mathcal{R}_1 = \cup_{j=1}^{n}\{PK_{u_j}^{RS}\}$ from $\mathcal{CH}$. $\mathcal{B}$ then chooses an Identity Based Signcryption scheme IBSC whose public parameters $Params^{IBSC}$ are independently generated from the public parameters of RS. $\mathcal{B}$ then sends $\mathcal{R} = (\cup_{j=1}^{n}\{PK_{u_j}^{RS}, ID_{u_j}\})$ to $\mathcal{A}$.

We now show how requests from $\mathcal{A}$ are treated by $\mathcal{B}$ who plays the role of challenger to $\mathcal{A}$.

- OSign queries: For i^{th} query on inputs $(m^{(i)}, \mathcal{R}^{(i)}, u_s^{(i)}, u_d^{(i)})$ from $\mathcal{A}$ where $\mathcal{R}^{(i)} \subseteq \mathcal{R}$, $\mathcal{B}$ gives $(m^{(i)}, \mathcal{R}_1^{(i)}, u_s^{(i)})$ to $\mathcal{CH}$ which after running $\sigma_1^{(i)} \leftarrow \text{RSign}(m^{(i)}, \mathcal{R}^{(i)}, SK_{u_s^{(i)}}^{RS}, Params^{RS})$, returns $\sigma_1^{(i)}$ to $\mathcal{B}$. Then, $\mathcal{B}$ runs $C^{(i)} \leftarrow \text{Signcrypt}((m^{(i)}||\sigma_1^{(i)}||\mathcal{R}^{(i)}), SK_{u_s^{(i)}}^{IBSC}, ID_{u_d^{(i)}}, ID_{u_s^{(i)}}, Params^{IBSC})$. Strings $r_1^{(i)}, \ldots, r_z^{(i)}$ are chosen uniformly at random by $\mathcal{B}$ and then $\sigma_2^{(i)} \leftarrow C^{(i)} \oplus H_1(r_1^{(i)}|| \ldots ||r_z^{(i)})$ is computed. $\mathcal{B}$ then computes $\sigma_3^{(i)} = H_2(r_1^{(i)}|| \ldots ||r_z^{(i)})$ and finally returns $(\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}), (r_1^{(i)}, \ldots, r_z^{(i)}))$ to $\mathcal{A}$.

Once this stage is over, $\mathcal{A}$ outputs $(m^*, \sigma^*, \mathcal{R}^*)$ to $\mathcal{B}$ which then $\mathcal{B}$ outputs the same to $\mathcal{CH}$. It is clear from the simulated game above that whenever σ^* is a forged valid signature on message m^* and ring $\mathcal{R}^*$ for our proposed RS-DVSI scheme, the same is a forged valid signature on the same message m^* and the ring $\mathcal{R}^*$ for the ring signature scheme RS also. Hence, $\mathcal{A}dv(\mathcal{B}) \geq \mathcal{A}dv(\mathcal{A})$.

Theorem 16 *The Proposed scheme preserves the uniqueness of the signer and the designated verifier if H_2 is collision resistant and **IBSC** is identity collision-resistant.*

Proof : Let $\mathcal{A}$ produce two tuples $(m, \sigma = (\sigma_1, \sigma_2, \sigma_3), \mathcal{R}, u_{s_0}, u_{d_0}, (r_{1,1}, \ldots, r_{1,z}))$ and $(m, \sigma = (\sigma_1, \sigma_2, \sigma_3), \mathcal{R}, u_{s_1}, u_{d_1}, (r'_{1,1}, \ldots, r'_{1,z}))$ to $\mathcal{B}$ where $\{u_{s_0}, u_{d_0}\} \neq \{u_{s_1}, u_{d_1}\}$. Then the following cases arise:

1. If $(r_{1,1}, \ldots, r_{1,z}) \neq (r'_{1,1}, \ldots, r'_{1,z})$, then $\mathcal{B}$ can find collision on H_2 as $H_2((r_{1,1}, \ldots, r_{1,z}) = H_2(r'_{1,1}, \ldots, r'_{1,z}) = \sigma_3$.

2. If $(r_{1,1}, \ldots, r_{1,z}) = (r'_{1,1}, \ldots, r'_{1,z})$, then $\mathcal{B}$ can find collision on the identities of **IBSC** as in this case Signcrypt$(m, SK^{IBSC}_{u_{s_0}}, ID_{u_{d_0}}, ID_{u_{s_0}}, Params^{IBSC}) =$ Signcrypt$(m, SK^{IBSC}_{u_{s_1}}, ID_{u_{d_1}}, ID_{u_{s_1}}, Params^{IBSC})$.

Theorem 17 *Let the ring signature scheme RS preserve unconditional anonymity. Then, if there exists an adversary $\mathcal{A}$ which can break the signer's privacy of the proposed RS-DVSI scheme, then there exists an adversary $\mathcal{B}$ which can break the ANON-IBSC-CCA security of the identity based signcryption scheme IBSC. Moreover, $\mathcal{A}dv(\mathcal{B}) = \mathcal{A}dv(\mathcal{A})/2$.*

Proof : Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBSC} and KeyGen^{IBSC} of identity based signcryption scheme IBSC. We shall show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to break the ANON-IBSC-CCA security of IBSC. $\mathcal{B}$ then chooses a ring signature scheme RS whose public parameters $Params^{RS}$ are independently generated from the public parameters of IBSC. $\mathcal{B}$ then runs KeyGen algorithm of RS for users $\cup_{j=1}^{n}\{u_j\}$. Let the output be $\mathcal{R}_1 = \cup_{j=1}^{n}\{PK^{RS}_{u_j}\}$. $\mathcal{B}$ stores the secret keys $\cup_{j=1}^{n}\{SK^{RS}_{u_j}\}$ in a list $\mathcal{L}$ which will be used during corrupt oracle queries. $\mathcal{B}$ then sends $\mathcal{R} = (\cup_{j=1}^{n}\{PK^{RS}_{u_j}, ID_{u_j}\})$ to $\mathcal{A}$.

We now show how requests from $\mathcal{A}$ are treated by $\mathcal{B}$ who plays the role of challenger to $\mathcal{A}$.

- Corrupt queries: For i^{th} query on $u_j^{(i)}$ from $\mathcal{A}$, $\mathcal{B}$ sends $ID_{u_j^{(i)}}$ to $\mathcal{CH}$ which in turn returns $SK^{IBSC}_{u_j^{(i)}}$ to $\mathcal{B}$. $\mathcal{B}$ searches in the list $\mathcal{L}$ for the secret key of $u_j^{(i)}$ and then returns $(SK^{RS}_{u_j^{(i)}}, SK^{IBSC}_{u_j^{(i)}})$ to $\mathcal{A}$.

- OSign queries: For i^{th} query on inputs $(m^{(i)}, \mathcal{R}^{(i)}, u_s^{(i)}, u_d^{(i)})$ from $\mathcal{A}$ where $\mathcal{R}^{(i)} \subseteq \mathcal{R}$, $\mathcal{B}$ runs $\sigma_1^{(i)} \leftarrow$ RSign$(m^{(i)}, \mathcal{R}^{(i)}, SK^{RS}_{u_s^{(i)}}, Params^{RS})$. $\mathcal{B}$ gives $m^{(i)}$, $\sigma_1^{(i)}$, $\mathcal{R}_1^{(i)}$, $u_s^{(i)}$ and $u_d^{(i)}$ to $\mathcal{CH}$ which after running $C^{(i)} \leftarrow$ Signcrypt$((m^{(i)}||\sigma_1^{(i)}||\mathcal{R}^{(i)}), SK^{IBSC}_{u_s^{(i)}}, ID_{u_d^{(i)}}, ID_{u_s^{(i)}}, Params^{IBSC})$ returns $C^{(i)}$ to $\mathcal{B}$. Strings $r_1^{(i)}, \ldots, r_z^{(i)}$ are chosen uniformly at random by $\mathcal{B}$ and then $\sigma_2^{(i)} \leftarrow C^{(i)} \oplus H_1(r_1^{(i)}|| \ldots ||r_z^{(i)})$ is computed. $\mathcal{B}$ then computes $\sigma_3^{(i)} = H_2(r_1^{(i)}|| \ldots ||r_z^{(i)})$ and finally returns $(\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}), (r_1^{(i)}, \ldots, r_z^{(i)}))$ to $\mathcal{A}$.

Once this stage is over, $\mathcal{A}$ submits $(m, \mathcal{R}', u_{s_0}, u_{s_1}, u_d)$ to $\mathcal{B}$ where $\mathcal{R}' \subseteq \mathcal{R}$. $\mathcal{B}$ then computes $\sigma_1 \leftarrow$ RSign$(m, \mathcal{R}, SK^{RS}_{u_{s_0}}, Params^{RS})$. $\mathcal{B}$ then sends $((m||\sigma_1||\mathcal{R}), \{u_{s_0}, u_{s_1}\}, \{u_{d_0}, u_{d_1}\})$ to $\mathcal{CH}$ where $d_0 = d_1 = d$. $\mathcal{CH}$ then chooses $b \in \{0,1\}$ and $b' \in \{0,1\}$ uniformly at random and computes $\mathcal{C}_{b,b'} \leftarrow$ Signcrypt$((m||\sigma_1||\mathcal{R}'), SK^{IBSC}_{u_{s_b}}, ID_{u_{d_{b'}}}, ID_{u_{s_b}}, Params^{IBSC})$. String $r_1, \ldots, r_z$ is chosen uniformly at random by $\mathcal{B}$ and then $\sigma_2 \leftarrow C_{b,b'} \oplus H_1(r_1)|| \ldots ||H_1(r_z)$ is computed. $\mathcal{B}$ then computes $\sigma_3 = H_2(r_1|| \ldots ||r_z)$ and returns $(\sigma = (\sigma_1, \sigma_2, \sigma_3), (r_1, \ldots, r_z))$ to $\mathcal{A}$.

$\mathcal{A}$ may perform queries on Corrupt and OSign with the restriction that Corrupt queries on u_{s_0}, u_{s_1} and u_d will be ignored by $\mathcal{B}$. $\mathcal{A}$ then submits a bit $\hat{b}$. $\mathcal{B}$ then chooses a bit $\hat{b'} \in \{0,1\}$ uniformly at random and outputs $(\hat{b}, \hat{b'})$ to $\mathcal{CH}$.

Analysis of the Proof.

We first show that the simulation is correct -

1. If the ring signature scheme RS is unconditionally anonymous, then the ring signature σ on the message m generated by the signer u_{s_0} can be obtained by the signer u_{s_1} also, if $\{PK_{u_{s_0}}, PK_{u_{s_1}}\} \subseteq \mathcal{R}$. Therefore, in the challenge phase, when $\mathcal{A}$ submits $(m, \mathcal{R}, u_{s_0}, u_{s_1}, u_d)$ to $\mathcal{B}$, the signer chosen by $\mathcal{B}$, in this case u_{s_0}, is same as u_{s_1}. Hence the bit b chosen by $\mathcal{CH}$ while signcrypting the message fits properly with the ring signature σ_1 generated by $\mathcal{B}$. Moreover, it can be easily observed that if RS in not unconditionally anonymous, signer's privacy can be broken by breaking the anonymity of RS only.

2. During the game, $\mathcal{B}$ is not allowed to query on the secret keys of u_{s_0}, u_{s_1} and u_d to $\mathcal{CH}$ as per the definition of ANON-IBSC-CCA game. Simultaneously, $\mathcal{A}$ also is not allowed to query on the secret keys of u_{s_0}, u_{s_1} and u_d to $\mathcal{B}$ as per the definition of signer's privacy. $\mathcal{B}$ queries for the secret keys on those identities which are queried by $\mathcal{A}$ only. Hence, $\mathcal{A}$ does not query for the secret key for u_{s_0}, u_{s_1} and u_d and so does $\mathcal{B}$.

3. Finally, $\mathcal{B}$ is not allowed to query for the designcryption of ciphertext $C_{b,b'}$ on the identity of $\{u_{s_0}, u_d\}$ or $\{u_{s_1}, u_d\}$ as per the definition of ANON-IBSC-CCA game. It is clear from the simulation that $\mathcal{B}$ never makes any designcryption query on any ciphertext and hence never makes designcryption query on the challenge ciphertext.

We now prove that $\mathcal{A}dv(\mathcal{B}) = \mathcal{A}dv(\mathcal{A})/2$. The bit guessed by $\mathcal{B}$, $\hat{b'}$, will be equal to b' with probability 1/2. Let,

E_1 denotes the event that $\mathcal{A}$ wins the game, i.e., $b = \hat{b}$.

E_2 denotes the event that $\mathcal{B}$ wins the game, i.e., $(b, b') = (\hat{b}, \hat{b'})$.

E_3 denotes the event that $b' = \hat{b'}$. $\Pr[b' = \hat{b'}] = 1/2$.

It is clear that $\Pr[E_2] = \Pr[E_1 \wedge E_3]$. Since, E_1 and E_3 are independent events, $\Pr[E_2] = \Pr[E_1 \wedge E_3] = \Pr[E_1]\Pr[E_3] = 1/2 \times \Pr[E_1]$.

Now, $\mathcal{A}dv(\mathcal{B}) = |\Pr[E_2] - 1/4| = |\Pr[E_1]/2 - 1/4| = \frac{1}{2}|\Pr[E_1] - 1/2| = \mathcal{A}dv(\mathcal{A})/2$.

Theorem 18 *If there exists an adversary $\mathcal{A}$ which can break the designated verifier's privacy of the proposed RS-DVSI scheme, then there exists an adversary $\mathcal{B}$ which can break the ANON-IBSC-CCA security of the identity based signcryption scheme IBSC. Moreover, $\mathcal{A}dv(\mathcal{B}) = \mathcal{A}dv(\mathcal{A})/2$.*

Proof : The proof goes in a similar manner as the proof goes for signer's privacy. Let there be a PPT challenger $\mathcal{CH}$ which runs the Setup^{IBSC} and KeyGen^{IBSC} of identity based signcryption scheme IBSC. We shall show how to construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to

break the ANON-IBSC-CCA security of IBSC. $\mathcal{B}$ then chooses a ring signature scheme RS whose public parameters $Params^{RS}$ are independently generated from the public parameters of IBSC. $\mathcal{B}$ then runs KeyGen algorithm of RS for users $\cup_{j=1}^{n}\{u_j\}$. Let the output be $\mathcal{R}_1 = \cup_{j=1}^{n}\{PK_{u_j}^{RS}\}$. $\mathcal{B}$ stores the secret keys $\cup_{j=1}^{n}\{SK_{u_j}^{RS}\}$ in a list $\mathcal{L}$ which will be used during corrupt oracle queries. $\mathcal{B}$ then sends $\mathcal{R} = (\cup_{j=1}^{n}\{PK_{u_j}^{RS}, ID_{u_j}\})$ to $\mathcal{A}$.

We now show how requests from $\mathcal{A}$ are treated by $\mathcal{B}$ who plays the role of challenger to $\mathcal{A}$.

- Corrupt queries: For i^{th} query on $u_j^{(i)}$ from $\mathcal{A}$, $\mathcal{B}$ sends $ID_{u_j^{(i)}}$ to $\mathcal{CH}$ which in turn returns $SK_{u_j^{(i)}}^{IBSC}$ to $\mathcal{B}$. $\mathcal{B}$ searches in the list $\mathcal{L}$ for the secret key of $u_j^{(i)}$ and then returns $(SK_{u_j^{(i)}}^{RS}, SK_{u_j^{(i)}}^{IBSC})$ to $\mathcal{A}$.

- OSign queries: For i^{th} query on inputs $(m^{(i)}, \mathcal{R}^{(i)}, u_s^{(i)}, u_d^{(i)})$ from $\mathcal{A}$ where $\mathcal{R}^{(i)} \subseteq \mathcal{R}$, $\mathcal{B}$ runs $\sigma_1^{(i)} \leftarrow \text{RSign}(m^{(i)}, \mathcal{R}^{(i)}, SK_{u_s^{(i)}}^{RS}, Params^{RS})$. $\mathcal{B}$ gives $m^{(i)}$, $\sigma_1^{(i)}$, $\mathcal{R}_1^{(i)}$, $u_s^{(i)}$ and $u_d^{(i)}$ to $\mathcal{CH}$ which after running $C^{(i)} \leftarrow \text{Signcrypt}((m^{(i)}||\sigma_1^{(i)}||\mathcal{R}^{(i)}), SK_{u_s^{(i)}}^{IBSC}, ID_{u_d^{(i)}}, ID_{u_s^{(i)}}, Params^{IBSC})$ returns $C^{(i)}$ to $\mathcal{B}$. Strings $r_1^{(i)}, \ldots, r_z^{(i)}$ are chosen uniformly at random by $\mathcal{B}$ and then $\sigma_2^{(i)} \leftarrow C^{(i)} \oplus H_1(r_1^{(i)}|| \ldots ||r_z^{(i)})$ is computed. $\mathcal{B}$ then computes $\sigma_3^{(i)} = H_2(r_1^{(i)}|| \ldots ||r_z^{(i)})$ and finally returns $(\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}), (r_1^{(i)}, \ldots, r_z^{(i)}))$ to $\mathcal{A}$.

Once this stage is over, $\mathcal{A}$ submits $(m, \mathcal{R}', u_s, u_{d_0}, u_{d_1})$ to $\mathcal{B}$ where $\mathcal{R}' \subseteq \mathcal{R}$. $\mathcal{B}$ then computes $\sigma_1 \leftarrow \text{RSign}(m, \mathcal{R}, SK_{u_s}^{RS}, Params^{RS})$. $\mathcal{B}$ then sends $((m||\sigma_1||\mathcal{R}), \{u_{s_0}, u_{s_1}\}, \{u_{d_0}, u_{d_1}\})$ to $\mathcal{CH}$ where $s_0 = s_1 = s$. $\mathcal{CH}$ then chooses $b \in \{0,1\}$ and $b' \in \{0,1\}$ uniformly at random and computes $\mathcal{C}_{b,b'} \leftarrow \text{Signcrypt}((m||\sigma_1||\mathcal{R}'), SK_{u_{s_b}}^{IBSC}, ID_{u_{d_{b'}}}, ID_{u_{s_b}}, Params^{IBSC})$. Strings $r_1, \ldots, r_z$ are chosen uniformly at random by $\mathcal{B}$ and then $\sigma_2 \leftarrow C_{b,b'} \oplus H_1(r_1)|| \ldots ||H_1(r_z)$ is computed. $\mathcal{B}$ then computes $\sigma_3 = H_2(r_1|| \ldots ||r_z)$ and returns $(\sigma = (\sigma_1, \sigma_2, \sigma_3), (r_1, \ldots, r_z))$ to $\mathcal{A}$.

$\mathcal{A}$ may perform queries on Corrupt and OSign with the restriction that Corrupt queries on u_s, u_{d_0} and u_{d_1} will be ignored by $\mathcal{B}$. $\mathcal{A}$ then submits a bit $\hat{b}'$. $\mathcal{B}$ then chooses a bit $\hat{b} \in \{0,1\}$ uniformly at random and outputs $(\hat{b}, \hat{b}')$ to $\mathcal{CH}$.

Analysis of the Proof.

We first show that the simulation is correct -

1. During the game, $\mathcal{B}$ is not allowed to query on the secret keys of u_s, u_{d_0} and u_{d_1} to $\mathcal{CH}$ as per the definition of ANON-IBSC-CCA game. Simultaneously, $\mathcal{A}$ also is not allowed to query on the secret keys of u_s, u_{d_0} and u_{d_1} to $\mathcal{B}$ as per the definition of designated verifier's privacy. Since $\mathcal{B}$ queries for the secret keys on those identities which are queried by $\mathcal{A}$ only. Hence, $\mathcal{A}$ does not query for the secret key for u_s, u_{d_0} and u_{d_1} and so does $\mathcal{B}$.

2. Finally, $\mathcal{B}$ is not allowed to query for the designcryption of ciphertext $C_{b,b'}$ on the identity of $\{u_s, u_{d_0}\}$ or $\{u_s, u_{d_1}\}$ as per the definition of ANON-IBSC-CCA game. It is clear from the simulation that $\mathcal{B}$ never makes any designcryption query on any ciphertext and hence never makes designcryption query on the challenge ciphertext.

We now prove that $\mathcal{A}dv(\mathcal{B}) = \mathcal{A}dv(\mathcal{A})/2$. The bit guessed by $\mathcal{B}$, $\hat{b}$, will be equal to b with probability 1/2. Let,

E_1 denotes the event that $\mathcal{A}$ wins the game, i.e., $b' = \hat{b}'$.

E_2 denotes the event that $\mathcal{B}$ wins the game, i.e., $(b, b') = (\hat{b}, \hat{b}')$.

E_3 denotes the event that $b = \hat{b}$. $\Pr[b = \hat{b}] = 1/2$.

It is clear that $\Pr[E_2] = \Pr[E_1 \wedge E_3]$. Since, E_1 and E_3 are independent events, $\Pr[E_2] = \Pr[E_1 \wedge E_3] = \Pr[E_1]\Pr[E_3] = 1/2 \times \Pr[E_1]$.

Now, $\mathcal{A}dv(\mathcal{B}) = |\Pr[E_2] - 1/4| = |\Pr[E_1]/2 - 1/4| = \frac{1}{2}|\Pr[E_1] - 1/2| = \mathcal{A}dv(\mathcal{A})/2$.

Chapter 7

Conclusion

This thesis proposes a new security notion called indistinguishable against chosen ciphertext verification attack (IND-CCVA), two new identity based signcryption schemes, two new signcryption schemes in public key setting and a new variant of ring signature called ring signature with designated verifier for signer's identity (RS-DVSI).

Chapter 3 proposes a new security notion called IND-CCVA and gives its formal definition. It then shows a gap between IND-CPA and IND-CCVA as well as a gap between IND-CCVA and IND-CCA. Further, it shows the existence of IND-CCA1 secure encryption scheme which is not IND-CCVA secure.

It could be interesting to see the other direction between IND-CCA1 and IND-CCVA. Moreover, from practical side, apart from Cramer-Shoup light version, the search of new and efficient IND-CCVA secure schemes particularly more efficient than IND-CCA secure encryption schemes could be a significant direction in this area.

Chapter 4 provides two different generic constructions for achieving IND-IBSC-CCA and ESUF-IBSC-CMA secure identity based signcryption schemes from IND-ID-CCA secure identity based encryption scheme and SUF-ID-CMA secure identity based signature schemes. The difference between these constructions lies in the computational efficiency. The first construction is efficient in signcrypt phase in which encrypt and sign algorithms can be run in parallel. Whereas the second construction is efficient in both signcrypt and designcrypt phases due to possible parallelisation of encrypt and sign algorithms during signcrypt phase and that of decrypt and verify algorithms during designcrypt phase.

The further direction in this area could be to find generic construction of IND-IBSC-CCA and ESUF-IBSC-CMA secure identity based signcryption schemes from less secure (than IND-ID-CCA and SUF-ID-CMA) identity based encryption and identity based signature schemes in the standard model.

Chapter 5 provides the efficient generic constructions of achieving dM-IND-iCCA secure signcryption schemes in the random oracle from (a) OWE-CPA secure and (b) OWE-PCA secure encryption schemes. OWE-CPA is the minimum security required for any public key

encryption scheme whereas dM-IND-iCCA is the highest known level of security for any signcryption scheme in public key setting. Both constructions give different but almost similar signcryption schemes which achieve the same level of security. Both these constructions are computationally efficient because encrypt and sign algorithms can be run in parallel during signcrypt phase as well as decrypt and verify algorithms can be run in parallel during designcrypt phase.

Proposed construction achieves the highest level of security of signcryption schemes from the least secure encryption schemes. The possibility to construct more efficient dM-IND-iCCA secure signcryption scheme from OWE-CPA secure encryption and that too in standard model would be an interesting problem for future in this area.

Chapter 6 proposes a new variant of Ring Signature called Ring Signature with Designated Verifier for Signer-Identity (RS-DVSI) and provides one generic construction. The appropriate motivation behind this variant can be phrased as "How to leak a secret and reap the reward too?". Proposed variant enables a user to leak a secret publicly and remain obscure in the future. User, if wishes, can then reveal its identity in the future to only a specified user termed as designated verifier; otherwise, it may remain obscure.

The proposed construction uses ring signature and identity based signcryption scheme to construct RS-DVSI scheme. There could be a possibility of different constructions from other cryptographic primitives or even more, less secure cryptographic primitives to achieve more efficient RS-DVSI schemes and thus worth exploring.

Bibliography

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n Signatures from a Variety of Keys. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 2002.

[2] W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr. RSA and Rabin Functions: Certain Parts are as Hard as the Whole. *SIAM Journal of Computing*, 17(2):194–209, April 1988.

[3] J.H. An, Y. Dodis, and T. Rabin. On the Security of Joint Signature and Encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[4] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems, Second Edition.* Wiley Publishing, Inc, 2008.

[5] N. Attrapadung, J. Furukawa, T. Gomi, G. Hanaoka, H. Imai, and R. Zhang. Efficient Identity-Based Encryption with Tight Security Reduction. In *Cryptology and Network Security*, volume 4301 of *Lecture Notes in Computer Science*, pages 19–36. Springer-Verlag, 2006.

[6] J. Baek, R. Steinfeld, and Y. Zheng. Formal Proofs for the Security of Signcryption. In *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer-Verlag, 2002.

[7] P.S.L.M. Barreto, B. Libert, N. McCullagh, and J.J. Quisquater. Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2005.

[8] M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *ACM Symposium on Theory of Computing*, STOC '98, pages 419–428. ACM, 1998.

[9] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.

[10] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 2000.

[11] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, CCS '93, pages 62–73. ACM, 1993.

[12] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1995.

[13] A. Bender, J. Katz, and R. Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. *Journal of Cryptology*, 22(1):114–138, 2009.

[14] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer-Verlag, 1998.

[15] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[16] D. Boneh, C. Gentry, and M. Hamburg. Space-Efficient Identity Based Encryption Without Pairings. In *IEEE Symposium on Foundations of Computer Science*, FOCS 2007, pages 647–657. IEEE Computer Society, 2007.

[17] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.

[18] X. Boyen. Multipurpose Identity-Based Signcryption - A Swiss Army Knife for Identity-Based Cryptography. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer-Verlag, 2003.

[19] E. Bresson, J. Stern, and M. Szydlo. Threshold Ring Signatures and Applications to Ad-hoc Groups. In *Advances in Cryptology - CRYPTO 2003*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer-Verlag, 2002.

[20] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. `http://eprint.iacr.org/2000/067`, 2000.

[21] R. Canetti and H. Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001.

[22] S. Chatterjee and P. Sarkar. Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In *Information Security and Cryptology - ICISC 2005*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer-Verlag, 2005.

[23] S. Chatterjee and P. Sarkar. *Identity-Based Encryption*. Springer, 2011.

[24] D. Chaum and E.V. Heyst. Group Signatures. In *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

[25] L. Chen and J. Malone-Lee. Improved Identity-Based Signcryption. In *Public Key Cryptography - PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 362–379. Springer-Verlag, 2005.

[26] D. Chiba, T. Matsuda, J.C.N. Schuldt, and K. Matsuura. Efficient Generic Constructions of Signcryption with Insider Security in the Multi-user Setting. In *Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 220–237. Springer-Verlag, 2011.

[27] J.C. Choon and J.H. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. In *Public Key Cryptography - PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2002.

[28] S.S.M. Chow, V.K.W. Wei, J.K. Liu, and T.H. Yuen. Ring Signatures without Random Oracles. In *ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, pages 297–302. ACM, 2006.

[29] C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, 2001.

[30] R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded CCA2-Secure Encryption. In *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 502–518. Springer-Verlag, 2007.

[31] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology - CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998.

[32] R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer-Verlag, 2002.

[33] A.W. Dent. Hybrid Signcryption Schemes with Insider Security. In *Information Security and Privacy*, volume 3574 of *Lecture Notes in Computer Science*, pages 253–266. Springer-Verlag, 2005.

[34] A.W. Dent. Hybrid Signcryption Schemes with Outsider Security. In *Information Security*, volume 3650 of *Lecture Notes in Computer Science*, pages 203–217. Springer-Verlag, 2005.

[35] A.W. Dent and Y.Zheng (editors). *Practical Signcryption*. Springer, 2010.

[36] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[37] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous Identification in Ad Hoc Groups. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer-Verlag, 2004.

[38] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[39] T. ElGamal. A PUBLIC KEY CRYPTOSYSTEM AND A SIGNATURE SCHEME BASED ON DISCRETE LOGARITHMS. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[40] E. Elkind and A. Sahai. A Unified Methodology For Constructing Public-Key Encryption Schemes Secure Against Adaptive Chosen-Ciphertext Attack. `http://eprint.iacr.org/2002/042`, 2002.

[41] M. K. Franklin and H. Zhang. A Framework for Unique Ring Signatures. `http://eprint.iacr.org/2012/577`, 2012.

[42] A.O. Freier, P. Karlton, and P.C. Kocher. The SSL Protocol. Version 3.0. `http://tools.ietf.org/search/draft-ietf-tls-ssl-version3-00`, 1996.

[43] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.

[44] W. Gao, G. Wang, X. Wang, and D. Xie. Controllable Ring Signatures. In *Information Security Applications*, volume 4298 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2007.

[45] C. Gentry. Practical Identity-Based Encryption without Random Oracles. In *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer-Verlag, 2006.

[46] O. Goldreich. A Uniform Complexity Treatment of Encryption and Zero-Knowledge. *Journal of Cryptology*, 6(1):21–35, 1993.

[47] O. Goldreich. *Foundations of Cryptography, Basic Applications*, volume 2. Cambridge University Press, 2004.

[48] S. Goldwasser, S. Micali, and P. Tong. Why and How to Establish a Private Code On a Public Network. In *23rd Annual Symposium on Foundations of Computer Science*, FOCS 1982, pages 134–144. IEEE Computer Society, 1982.

[49] S. Goldwasser and S. Micalli. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[50] J. Håstad and M. Näslund. The Security of Individual RSA Bits. In *39th Annual Symposium on Foundations of Computer Science*, FOCS 1998, pages 510–521. IEEE Computer Society, 1998.

[51] J. Herranz and G. Sáez. Forking Lemmas for Ring Signature Schemes. In *Progress in Cryptology - INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279. Springer-Verlag, 2003.

[52] D. Hofheinz and E. Kiltz. Secure Hybrid Encryption from Weakened Key Encapsulation. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 553–571. Springer-Verlag, 2007.

[53] S. Hohenberger, A. Lewko, and B. Waters. Detecting Dangerous Queries: A New Approach for Chosen Ciphertext Security. In *Advances in Cryptology - EUROCRYPT 2002*, volume 7237 of *Lecture Notes in Computer Science*, pages 663–681. Springer-Verlag, 2012.

[54] RSA Data Security Inc. PKCS #1: RSA Encryption Standard, Version 1.5. 1993.

[55] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated Verifier Proofs and Their Applications. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer-Verlag, 1996.

[56] M.P. Jhanwar and R. Barua. A Variant of Boneh-Gentry-Hamburg's Pairing-Free Identity Based Encryption Scheme. In *Information Security and Cryptology*, volume 5487 of *Lecture Notes in Computer Science*, pages 314–331. Springer-Verlag, 2009.

[57] E. Kiltz and Y. Vahlis. CCA2 Secure IBE: Standard Model Efficiency through Authenticated Symmetric Encryption. In *Topics in Cryptology - CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 221–238. Springer-Verlag, 2008.

[58] M. Klonowski, L. Krzywiecki, M. Kutylowski, and A. Lauks. Step-Out Ring Signatures. In *Mathematical Foundations of Computer Science 2008*, volume 5162 of *Lecture Notes in Computer Science*, pages 431–442. Springer-Verlag, 2008.

[59] K. Kurosawa and S. H. Heng. Identity-Based Identification Without Random Oracles. In *Computational Science and its Applications - ICCSA 2005*, volume 3481 of *Lecture Notes in Computer Science*, pages 603–613. Springer-Verlag, 2005.

[60] J.S. Lee and J. Chang. Strong Designated Verifier Ring Signature Scheme. In *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, pages 543–547. Springer, 2007.

[61] K.C. Lee, H.A. Wen, and T. Hwang. Convertible ring signature. *IEEE Proceedings of Communications*, 152(4):411–414, 2005.

[62] J. Li and Y. Wang. Universal Designated Verifier Ring Signature (Proof) Without Random Oracles. In *Emerging Directions in Embedded and Ubiquitous Computing*, volume 4097 of *Lecture Notes in Computer Science*, pages 332–341. Springer-Verlag, 2006.

[63] B. Libert. New Secure Applications of Bilinear Maps in Cryptography. In *Ph.D. Thesis, Catholic University, Louvain*, 2006.

[64] B. Libert and J. Quisquater. Efficient Signcryption with Key Privacy from Gap Diffie-Hellman Groups. In *Public Key Cryptography - PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer-Verlag, 2004.

[65] B. Libert and J. Quisquater. Improved Signcryption from q-Diffie-Hellman Problems. In *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 220–234. Springer-Verlag, 2004.

[66] J.K. Liu, V.K. Wei, and D.S. Wong. A Separable Threshold Ring Signature Scheme. In *Information Security and Cryptology - ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 12–26. Springer-Verlag, 2004.

[67] J.K. Liu, V.K. Wei, and D.S. Wong. Linkable Spontaneous Anonymous Group Signature for Ad Hoc Groups. In *Information Security and Privacy*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer-Verlag, 2004.

[68] J. Loftus, A. May, N. Smart, and F. Vercauteren. On CCA-Secure Somewhat Homomorphic Encryption. In *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 55–72. Springer-Verlag, 2011.

[69] T. Matsuda, K. Matsuura, and J.C.N.Schuldt. Efficient Constructions of Signcryption Schemes and Signcryption Composability. In *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 321–342. Springer-Verlag, 2009.

[70] R.J. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory. *DSN progress report*, 42(44):114–116, 1978.

[71] D. Naccache. Secure and practical identity-based encryption. *IET Information Security*, 1(2):59–64, 2007.

[72] M. Naor. Deniable Ring Authentication. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer-Verlag, 2002.

[73] M. Naor and M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *22nd Annual ACM Symposium on Theory of Computing*, STOC 1990, pages 427–437. ACM, 1990.

[74] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–174. Springer-Verlag, 2001.

[75] S.K. Pandey and R. Barua. Construction of Identity Based Signcryption Schemes. In *Information Security Applications*, volume 6513 of *Lecture Notes in Computer Science*, pages 1–14. Springer-Verlag, 2011.

[76] S.K. Pandey and R. Barua. Efficient Construction of Identity Based Signcryption Schemes from Identity Based Encryption and Signature Schemes. *Journal of Internet Services and Information Security*, 1(2/3):161–180, 2011.

[77] S.K. Pandey, S. Sarkar, and M.P. Jhanwar. Relaxing IND-CCA: Indistinguishability against Chosen Ciphertext Verification Attack. In *Security, Privacy, and Applied Cryptography Engineering*, volume 7644 of *Lecture Notes in Computer Science*, pages 63–76. Springer-Verlag, 2012.

[78] K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[79] M. Prabhakaran and M. Rosulek. Homomorphic Encryption with CCA Security. `http://eprint.iacr.org/2008/079`, 2008.

[80] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1991.

[81] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[82] R.L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565. Springer-Verlag, 2001.

[83] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairing. In *SCIS*, 2000.

[84] V. Saraswat, and S.K. Pandey. How to Leak a Secret and Reap the Rewards too. In *LATINCRYPT 2014*, to appear.

[85] A. Shamir. Identity Based Cryptosystems and Signature Schemes. In *Advances in Cryptology - CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.

[86] V. Shoup. Why Chosen Ciphertext Security Matters. Technical Report RZ 3076, IBM Zurich. 1998.

[87] D.R. Stinson. *CRYPTOGRAPHY - THEORY AND PRACTICE*. Chapman & Hall/CRC, 2006.

[88] C.H. Tan. Signcryption Scheme in Multi-user Setting without Random Oracles. In *Advances in Information and Computer Security*, volume 5312 of *Lecture Notes in Computer Science*, pages 64–82. Springer-Verlag, 2008.

[89] S. Vivek, S. Deva Selvi, and C. Pandu Rangan. CCA Secure Certificateless Encryption Schemes based on RSA. In *Security and Cryptography - SECRYPT 2011*, pages 208–217. SciTePress, 2011.

[90] S. Vivek, S. Deva Selvi, and C. Pandu Rangan. Stronger Public Key Encryption Schemes Withstanding RAM Scraper Like Attacks. `http://eprint.iacr.org/2012/118`, 2012.

[91] B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer-Verlag, 2005.

[92] J. Xu, Z. Zhang, and D. Feng. A Ring Signature Scheme Using Bilinear Pairings. In *Information Security Applications*, volume 3325 of *Lecture Notes in Computer Science*, pages 160–169. Springer-Verlag, 2004.

[93] F. Zhang and K. Kim. ID-Based Blind Signature and Ring Signature from Pairings. In *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer-Verlag, 2002.

[94] Y. Zheng. Digital Signcryption or how to achieve Cost(Signature & Encryption) << Cost(Signature)+Cost(Encryption). In *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.