Security in signcryption scheme

Kamil Ismayil

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2010

Avdeling for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Security in Signcryption scheme

Kamil Ismayil

2010/12/01

Abstract

Signcryption is a new cryptographic technology which provides both confidentiality and encryption. Before signcryption, confidentiality and encryption were considered separately. It means that encryption provided confidentiality and signature provided authenticity. In 1997, Zheng combined those two goals in one primitive scheme. By that way he was able to reduce computation and cost. After the primitive version of signcryption proposed by Zheng, many other types of signcryption were proposed.

In this thesis, I will analyze Zheng's signcryption scheme in different models proposed by other researchers. But initially we begin with a formal definition of signcryption, continuing with security models and their security issues. Then we present the advantages and disadvantages of those signcryption schemes and will analyze hybrid model of signcryption scheme. Finally, we will write future progress that might be done in signcryption.

The thesis is theoretical in nature, there is no experimental work.

Acknowledgements

I would like to thank my supervisor, Professor Slobadan Petrovic, for his help throughout this thesis. Thanks for all of his advices, numerous comments, suggestions, and corrections during the writing of this thesis. I would like to thank to all of my friends for collabaration and making the thesis better. Finally, I want to say thanks to my family for giving amazing support.

Table of Contents

List of Listings

List of Figures

List of Tables

Acronyms

KEM	Key Encapsulation Mechanism
DEM	Data Encapsulation Mechanism
GDH	Gap Diffie Hellman
GDL	Gap Discrete Log
IND	Indistinguishability
INT	Integrity
INP	Input
sUF	Strong Unforgeability
CCA	Chosen Ciphertext Attack
CPA	Chosen Plaintext Attack
CMA	Chosen Message Attack
ROM	Random Oracle Model
IND-CPA	Indistinguishability Under Chosen Plaintext Attacks
EXP	Modular exponentitations
MUL	Modular multiplications
DIV	Modular divisions
ADD	Modular addition or subtraction

Symbols

=	Equality
≠	Not equal
≈	Approximately
/	Divides
×	Multiplication
|p|	Number of bits in p
hash()	One way hash function
≤	Less than equal to
≥	Bigger then equal to
Z_p	set of integers mod p
Dec(c)	Decryption algorithm
Enc(m)	Encryption algorithm
SC()	Signcryption algorithm
USC()	Unsigncryption algorithm
Sign()	Signature algorithm
Ver()	Verification algorithm
G	Group
mod	modulo operator
⊤	Success
⊥	Failure
x	Public key
y	Private key

1 Introduction

1.1 Topics covered

First of all, basic preliminaries are going to be introduced. Here includes notation and terminology, public key encryption schemes, games, hash functions and random oracle models. Then I will introduce the main topic, signcryption. The main target is to present different types of signcryption schemes. Here includes both standard models and KEM/DEM framework. While introducing different types of signryption schemes, I will analyse their security notions which is the main point of this thesis. Next, Zheng signcryption scheme will be discussed in the term of authenticity and privacy. Finally, we will conclude the thesis by shortly analyzing Zheng's signcryption scheme in the hybrid model.

1.2 Keywords

Authenticity, privacy, algorithms, games, security models, encryption, signature, signcryption, hybrid signcryption, hash function.

1.3 Problem description

Encryption and signature schemes are very important in cryptology in terms of providing privacy and authenticity. Some decades ago a number of schemes emerged in order to meet privacy and authenticity requirements of the schemes separately. However there are many applications where both privacy and authenticity needs to be provided. In 1997 Zheng integrated both digital signature and public key encryption schemes and called it signcryption. In the thesis, we provide precise security models for Zheng signcryption scheme and security issues in the sense of confidentiality and unforgeability.
Providing confidentiality is a key issue in terms of security in cryptography. Note that, confidentiality is achieved by applying encryption in signcryption.
Another main property in signcryption is message authenticity which is achieved by digital signature. The main requirement in digital signature is unforgeability against message attacks (the adversary is allowed to send queries to random oracle model). Let's note that Schnorr and Elgamal signature schemes were proved being secure in the sense of unforgeability by doing a little change on them.
In previous schemes, the message and the sender's signature are achieved after decrypting the encrypted message. Then using the sender's public key the signature is being verified. Thus, anyone who knows the sender's public key can verify the message. But, in signcryption in order to verify the signature the receiver has to use his/her own private key.

1.4 Motivation

Signcryption is a new scheme which can work both as a signature scheme and encryption scheme. Because of the low cost and the efficiency, signcryption is widely used in many applications requring secure and authenticated message delivery. In the symmetric setting, some methods are considered in such internet standards as IPSec and SSL. More information about these methods can be found in [4], [14]. In asymmetric setting, signcryption introduced by Zheng provides confidentiality and unforgeability.

It is very important to continue more detailed analysis in signcryption in order to find the best models.

The motivation behind this thesis is to show different security models for signcryption scheme and discuss additional security goals. We conclude that signcryption satisfies security requirements in the respect of confidentiality and unforgeability.

1.5 Research questions

Throughout the thesis we answer the following questions:

1. Is it possible to construct a scheme by combining both encryption and signature features?
2. How much confidentiality is provided in Zheng's signcryption scheme?
3. How much unforgeability is provided in Zheng's signcryption scheme?
4. How does signcryption scheme look like in the hybrid model?

1.6 Contributions

The contribution of the thesis is in analyzing the Zheng signcryption scheme. The main contribution is to compare Zheng signcryption scheme in two different models and choose the most appropriate model.

1.7 Related work

In 1997 Zheng brought a new idea with his article "*Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption)*" [15]. In this article Zheng shows whether it is possible to perform a secure message transfer more efficiently. He was the first person who examined such schemes considering signcryption as a cryptographic primitive. In 2002, Baek, Steinfeld and Zheng [6] proved that, Zheng's scheme is secure enough and has well defined security properties.
In [6] Steinfeld, Baek and Zheng provided the first formal analysis of signcryption schemes. They constructed a new signryption scheme provided a security model for their own scheme. In addition, they proved the schemes unforgeability too. But unfortunately they did not provide confidentiality of the signcryption scheme.

A signcryption introduced by Zheng is very primitive. After Zheng, many work has done on signcryption. For example, An et al. introduced a proper model for signcryption, Dodis et al. offered a security in multi-user environment. For construction of hybrid signcryption, Dent and Bjorstad presented KEM/DEM framework. Let's note that, tag-KEM/DEM was introduced for construction of hybrid encryption. The main purpose of framework is to combine tag-KEM and DEM. A tag-KEM uses asymmetric technique to encrypt a symmetric key along with a tag, while the DEM uses a symmetric cipher to encrypt the message payload using the key from the KEM. The security of signcryption tag-KEM is defined for the security of signcryption which restricts so that the adversary is allowed to access de-signcryption oracle (resp. signcryption oracle) for the attacked user but not signcryption oracle (resp. de-signcryption oracle) for the attacked user if the adversary attacks privacy (resp. authenticity). On the other hand, the modified definition allows the adversary to access both oracles. It means that the modified definition for security of signcryption becomes stronger than the previous one and the security of signcryption tag-KEM in is not enough strong for yielding the modified security of signcryption.

Bellare and Namprepre [31] proposed security models based on symmetric encryption and message authentication. They proved that 'Encrypt-then-MAC' is much secure against ciphertext attacks and almost unforgeable. Krawczyk [32] applied special techniques to build secure channels over insecure networks. He proved that 'MAC-then-Encrypt' was secure too.

1.8 My contribution

The main purpose of this thesis is to propose a precise security model for signcryption and discuss security issues based on the proposed model. A main contribution is to provide the best security issues in the signcryption scheme. In order to do that I will analyze common attacks such as random oracle attacks. Confidentiality and integrity are the main considerations here. I will try to show that signcryption schemes are very resistant and almost unforgeable against message attacks. Additionally, for having high level of efficiency and security level I will do my analyses in hybrid signcryption scheme.

2 The whole chapter's preliminaries

2.1 Notation and terminology

Before beginning the thesis it is necessary to give information about notation, convention and terminology. In order to denote natural numbers, integers and real numbers N, Z and R are used respectively. To denote the set of binary strings $\{0,1\}^*$ will be used. While $\{0,1\}^n$ represents the set of binary strings of length n. Lets note that, |a| represents the length of a finite string a. The concatenation of two binary strings a and b is denoted as a||b. Pr[E] denotes the probability of an event R_E. While the negation of Pr[E] is being represented by Pr[¬E].

In this thesis there are a number of algorithms defined. The output of this algorithm is either $\top$ or $\perp$ respectively to denote success or failure. Let's guess that A is a *deterministic* algorithm with input I and output x. In order to denote this the notation x ← A(I) will be used. On the other hand, if algorithm A is *probabilistic* then the notation will be $x \xleftarrow{R} A(I)$.

The purpose of oracle is to give an algorithm access to good-designed computational power. By using oracle, the algorithm can get the result of calculations, although information that it has is not helpful to obtain the result alone. As an example, the value of a secret key can be given. The algorithm A that has input I and has access to an oracle O, is represented as A(I;O).

2.2 Public Key Encryption

In this section three public-key methods will be discussed.

Identification Scheme. The most important algorithm in identification scheme is Key_{ID}. Input to this algorithm is k and output is (x,y) (private key, public key). By using an identification scheme, one side (Linda) can identify herself to another part (John) without giving more detailed information. If the scheme is secure, then John can't get any information about Linda's private key.

Signature Scheme. There are three algorithms in a signature scheme: key generation, signature and verification algorithms. Input to the key generation algorithm is k and output is (x_s, y_s). It can be denoted as $(x_s, y_s) \xleftarrow{R} Key_s(k)$. Inputs to the signature algorithm are m and x_s, and output is σ (a signature). It can be denoted as $\sigma \xleftarrow{R} \text{Sign}(x_s, m)$. Verification algorithm's inputs are y_s, m and σ, while output is either $\top$ or $\perp$. It can be denoted as $\{\top,\perp\} \leftarrow \text{Ver}(y_s, m, \sigma)$.

Encryption Scheme. An encryption scheme consists of three algorithms: a key generation algorithm, encryption and decryption. Key generation algorithm can be denoted as $(x_r, y_r) \xleftarrow{R} Key_E$. Its input is k and output is (x_r, y_r). The encryption algorithm has input y_r and m , and outputs c: $c \xleftarrow{R} Enc(y_r, m)$. The input to the decryption algorithm is x_r and c, while output is m: $m \leftarrow \text{Dec}(x_r, c)$.

2.3 Games

In order to consider security notions it is important to introduce games. There are two participating parts in games: tester and attacker. The purpose of the games is that the tester has to test attacker in different security models under different circumstances. There are two different games introduced under two security models, IND-CCA2 and UF-CMA.

Public information:
Security parameter k
Encryption scheme: E= {Key_E, Enc, Dec}
IND-CCA2-experiment:
$(x_r, y_r) \overset{R}{\leftarrow} Key_E$(k)
$(m_0, m_1, state) \overset{R}{\leftarrow} A_1$(I, y_r, Dec_Oracle)
b $\overset{R}{\leftarrow}$ {0,1}
$c^* \overset{R}{\leftarrow}$ Enc(y_r, m_b)
$b' \overset{R}{\leftarrow} A_2$(I, y_r, state, c^*, Dec_Oracle)
A wins the game if b = b' and c^* has not been asked to Dec_Oracle.
Dec_Oracle(c):
Return Dec(x_r, c)

Listing 1: IND-CCA2 game for encryption scheme

Public Information:
Signature scheme S = {Key_S, Sign, Ver}
Security parameter k
UF-CMA experiment:
$(x_r, y_r) \overset{R}{\leftarrow} Key_S$(k)
$(m^*, s^*) \overset{R}{\leftarrow} A$(I, y_s, Sign_Oracle)
A wins if $\top \leftarrow$ Ver(y_s, m^*,s^*) and m^* has not been asked to Sign_Oracle
Sign_Oracle(m):
Return Sign(x_s, m)

Listing 2: UF-CMA game for signature scheme

2.4 Hash functions and ROM

Cryptographic hash functions
Cryptographic hash functions are very important in signcryption. A large set of elements is mapped to a smaller set of elements and can be denoted as h : X → Y.
Definition (Cryptographic Hash Function). A hash function has the following properties:

- Hash function(H) takes a message (m) of finite length and maps it to a fixed length output. The result is H(m).
- When m ϵ {0,1}* and H are known, it is not difficult to calculate H(m).

Hash functions are used to check integrity of PKE schemes and digital signature schemes. In signature schemes instead of signing the whole message, one can calculate hash value of the message and sign it.

Random Oracle Model. The availability of primitives such as SHA-1 [16] and MD5 [11] were suggested by Bellare and Rogaway [12]. But unfortunately the properties of these primitives had never been given formally. But they suggested ROM (random oracle model) where such primitives used to design provably-secure cryptosystems. The idea behind ROM is that, all parties have access to random oracle. Random oracle replies with a random response, when it is queried. In the beginning it is supposed that, an adversary must exploit the weakness in the hash function, if the protocol proved being secure. But then Canetti et al. [17] proved that this is not true. He proved that a cryptosystem should be secure in ROM.

2.5 GDH Problem

In order to prove the confidentiality of Zheng's signcryption scheme GDH problem needs to be reviewed. This problem is also called dual inverting and decisional problem, because an attacker tries to solve the problem with an oracle that solves a related decisional problem [6]. In GDH problem the attacker has access to O^{DDH} (Decisional Diffie Helman Oracle) that given ($\overline{g}$, $\overline{g}^u$, $\overline{g}^v$, z) ϵ (g)X(g)X(g)X(g) checks whether z = $\overline{g}^{uv}$ or not, tries to find Diffie-Hellman key K=g^{ab} corresponding to the given pair (g^a, g^b) [6].

Let's have a look to the game given below.

GDHGame(k,A^{GDH})
(g,p,q) ← GC(k)
a $\overset{R}{\leftarrow} Z_q^*$; b $\overset{R}{\leftarrow} Z_q^*$
K ← A^{GDH}((g,p,q),g^a, g^b | $O^{DDH}(\cdot,\cdot,\cdot,\cdot)$)
If K = g^{ab} then Return 1 else Return 0

where |p|=k, q|(p-1), q > $2^{l_q(k)}$, l_q: N → N is for determining the length of q, A^{GDH} is an attacker. O^{DDH} is DDH oracle which on input has ($\overline{g}$, $\overline{g}^u$, $\overline{g}^v$, z) and outputs 1 if z = $\overline{g}^{uv}$ and 0 otherwise.

The probability of the attacker's being successful is given below:

$$Succ^{GDH}_{A^{GDH},Z_p^*}(k) \overset{\text{def}}{=} \Pr[\text{GDHGame}(k,A^{GDH}) = 1]$$

2.6 GDL Problem

In order to prove the unforgeability of Zheng's signcryption scheme the GDL problem needs to be reviewed. Actually GDL is the discrete log analogue of the GDH problem and is easier than the classical discrete log problem, because the attacker has access to a restricted DDH oracle O^{rDDH}. The purpose of O^{rDDH} is to check whether z = ($\overline{g}^v$)^a or not with the given variables (g, g^a, $\overline{g}^v$, z) ϵ (g)X(g)X(g)X(g).

Let's have a look to the game given below:

GDLGame(k, A^{GDL})
(g,p,q) ← GC(k)
a $\xleftarrow{R}$ Z_q^*;
$a' \leftarrow A^{GDL}$((g,p,q),g^a | O^{rDDH}(g, g^a, ·, ·))
If a' = a then Return 1 else Return 0

where |p|=k, q|(p-1), q > $2^{l_q(k)}$, l_q: N → N is for determining the length of q, A^{GDH} is an attacker. O^{rDDH}(g, g^a, ·, ·) is a restricted DDH oracle which on input has (g, g^a, $\overline{g}^v$, z) and output 1 if z = $(\overline{g}^v)^a$ and 0 otherwise.

The probability of the attacker's being successful is given below:

$$Succ^{GDH}_{A^{GDH},Z_p^*}(k) \stackrel{\text{def}}{=} \Pr[\text{GDLGame}(k,A^{GDL}) = 1]$$

Lets note that, the attacker's purpose in GDH is weaker than in GDL. Precisely, the attacker's goal is to find K = g^{ab} corresponding to (g^a, g^b) keys. By using discrete log of g^a the attacker can calculate Diffie-Hellman key. It means that, if the attacker can solve the GDL problem, then he/she can also easily solve the GDH problem.

2.7 About Signcryption

Encryption and digital signatures are two main issues in public key cryptology. Until 1990s no one considered to combine these two features. But an obvious approach is to combine both encryption and digital signature sequentially. Nevertheless, this composition is not so efficient.

Confidentiality, integrity, non-repudiation and authentication are important issues in information security. In order to achieve these security services in many applications both encryption and digital signature are required. In public key encryption, a message is digitally signed and then followed by an encryption. It is called signature-then-encryption. In signature-then-encryption there are two problems: high cost and low efficiency. In order to solve the problems that exist in signature-then-encryption, the signcryption was offered. The signcryption is very new technique which is supposed to fulfill the functionalities of digital signature and encryption. First, the signcryption was offered by Zheng in 1997 [15]. SignCryption is a new concept in PKC (Public key cryptology). It provides a common framework for a number of protocols which are used to provide a confidential and authenticated transmission channel for messages. One of the best properties of signcryption is capability of providing both encryption and digital signature at the same time. In other words, by using SignCryption we can acquire confidentiality, authentication, integrity, unforgeability, non-repudiation, public verifiability and forward secrecy of message confidentiality. SignCryption provides not only low computational cost but also reducing communication overhead.

There were many signcryption schemes offered throughout the years. Each signcryption scheme has its own advantages, disadvantages and limitations. In a signcryption scheme, the sender uses the public key of the receiver (for forming a session key of symmetric encryption). On the other hand, the receiver uses his/her private key in

order to derive the same session key. Let's note that, capturing of session keys can be very dangerous for the security of the system.

2.8 Hybrid cryptography

Hybrid cryptography is much more advantageous than other constructed crypto schemes using symmetric or asymmetric techniques alone. That is because hybrid cryptology is concerned with the combination of keyed symmetric and asymmetric schemes. As a general rule, hybrid cryptology is used to create asymmetric encryption schemes where message encryption is provided by a symmetric encryption scheme, for instance AES in CBC mode, (under a randomly generated symmetric key). Then, the asymmetric encryption scheme is used to encrypt randomly generated symmetric key, which allows the asymmetric encryption scheme to handle long messages.

One of the best ways to handle large messages is to use hybrid cryptography. The hybrid cryptosystem can be constructed by using two different cryptosystems:

- KEM (Key encapsulation mechanism)
- DEM (Data encapsulation mechanism)

KEM is constructed by using public-key cryptosystem, while symmetric-key cryptosystem is being using to construct DEM. Note that for long messages the bulk of the work in encryption/decryption is done by the more efficient symmetric-key scheme, while the inefficient public-key scheme is used only to encrypt/decrypt a short key value. Such kind of hybrid cryptographic schemes has been much more attractive in the recent years.

2.9 Definitions of hybrid signcryption blocks

In this section I explain KEM, DEM, PKE which are main concepts in hybrid signcryption. Additionally possible attacks and their security consequences will be explained too.

2.9.1 DEM

DEM is a symmetric encryption scheme which consist of two algorithms: DEM Encryption(DEM.Enc) and DEM Decryption (DEM.Dec). DEM is associated to a key-space and message space defined by λ. We accept that key space is $\{0,1\}^{\lambda}$ and message space is $\{0,1\}^{*}$.

c ← DEM.$Enc_{dk}(m)$	DEM encryption algorithm encrypts a message m using symmetric key dk. The output is ciphertext c
m ← DEM. $Dec_{dk}(c)$	DEM decryption algorithm decrypts ciphertext c and recovers the original message m

2.9.2 PKE

Public key encryption consists of three algorithms: PKE generation, PKE encryption and PKE decryption.

(y,x) ← PKE.Gen(1^{λ})	A probabilistic algorithm that on input gets the security parameter, generates public and private keys (y,x). The public-key defines the message space M.
c ← $PKE.Enc_y(m)$	A probabilistic algorithm that encrypts a message $m \in M$ and the result is ciphertext c.
d ← $PKE.Dec_x(c)$	An algorithm that decrypts c and outputs $m \in M$.

2.9.3 KEM

KEM also consists of three algorithms: KEM generation, KEM encryption and KEM decryption.

(y,x) ← KEM.Gen(1^{λ})	A probabilistic algorithm that generates public and private keys (y x). The blpuic-key defines the key space $\mathcal{K}_k$.
(K, ϕ) ← KEM. Enc_y ()	A probabilistic algorithm that generates key $K \in \mathcal{K}_k$ and its encryption ϕ.
$K \leftarrow KEM.Dec_{sk}(\phi)$	An algorithm that decrypts ϕ to recover K.

2.9.4 Tag-KEM

KEM consists of 3 algorithms: key generation, encryption and decryption. As a general rule, the encryption function is divided into 2 parts: the first part selects a random key, while the second part encrypts the selected key with a given tag. The formal description of Tag-KEM is given below:

(y,x) ← TKEM.Gen(1^{λ})	A probabilistic algorithm that generates the public-key y and private-key x. The public-key defines all relative spaces, i.e., spaces for tags and encapsulated keys denoted by T and $\mathcal{K}_k$.
(ω, dk) ← TKEM.Key(y)	A probabilistic algorithm that outputs one-time key $dk \in \mathcal{K}_D$ and internal state information ω. $\mathcal{K}_D$ is the key-space of DEM.
Ψ ← TKEM.Enc(ω, T)	A probabilistic algorithm that encrypts dk (embedded in ω) into Ψ along with T, where T is called a tag.
$dk \leftarrow TKEM.Dec_x(\Psi, T)$	A decryption algorithm that recovers dk from Ψ and T. For soundness, $TKEM.Dec_{sk}(\Psi, T) = dk$ must hold for any x, dk, Ψ and T, associated by the above three functions.

Note that Tag-KEM is a common way of KEM if the tag is a fixed string.

Additionally, we give some definitions that are going to be used in the rest of the thesis.

Defnition (Negligible function) A function $f : Z \rightarrow R$ is negligible if for every polynomial p there exists an integer N_p such that $f(n) \leq 1/p(n)$ for all $n \geq N_p$.

Definition (IND security) A signcryption scheme is said to be IND-CCA secure if, for every polynomial-time attacker A, the advantage that A has in winning the IND-CCA game is negligible as a function of the security parameter k.

Definition (INP security) A signcryption KEM is INP-CCA secure if the attacker's advantage in winning the INP-CCA game is negligible as a function of the security parameter k.

Definition (INT security) A signcryption KEM is INT-CCA secure if the probability that attacker wins the integrity game is negligible as a function of the security parameter k.

Definition (sUF security) A signcryption scheme is said to be sUF-CCA secure if the probability that attacker wins the sUF-CCA game is negligible as a function of the security parameter k.

Definition (Insider security) A signcryption scheme is said to be insider secure if it is both IND-CCA secure and sUF-CCA secure.

Definition (Outsider security) A signcryption scheme is said to be outsider secure if it is both IND-CCA and sUF-CCA secure.

Definition (INT-CCA security) A signcryption DEM is said to be INT-CCA secure if the probability that attacker wins the INT-CCA game is negligible as a function of the security parameter k.

3 Encryption, Signature and Signcryption

3.1 Encryption Schemes

An encryption scheme is a triple of algorithms, E= (Enc-KeyGen, Enc, Dec):

- Enc-Keygen is an algorithm where the input is 1^k, k is a security parameter, and output is (y,x). y is a public key, while x is a private key. This operation can be denoted as (y,x) ← Enc-KeyGen(1^k).
- Enc is an algorithm where the input is y, a message m and the output is a ciphertext c. This operation can be shown as either c← $Enc_y(m)$ or c←Enc(m).
- Dec is an algorithm where an input is x and a ciphertext c, an output is a message m. The output can be $\perp$ if the ciphertext is wrong. This operation can be denoted as m← $Dec_{EK}(c)$ or m← Dec(c).

Security in Encryption Schemes

There are two main issues in the security of an encryption scheme:

- the goal of the adversary
- capabilities given to the adversary

The highest level of security is achieved at the time when adversary can't achieve the goal even with the strongest capabilities.

In order to understand the security in encryption scheme better, we should look at game theory. The actions that the adversary needs to follow can be written in the following way:

$$(o_1, o_2, \ldots\ldots, o_m) \leftarrow A\ (i_1, i_2, \ldots\ldots, i_n)$$

where A is the adversary, $o_1, o_2, \ldots\ldots, o_m$ are outputs, $i_1, i_2, \ldots\ldots, i_n$ are inputs.

In encryption, the goal of the adversary is to distinguish ciphertexts. First of all, the adversary is given the encryption key. Then the adversary needs to select two equal length messages: m_0 and m_1. In the last stage, one of the messages randomly chosen and encrypted, and the adversary needs to correctly determine (or guess) which of the messages is encrypted. The steps are given below:

1. (y, x) ← Enc-KeyGen(1^k)
2. (m_0, m_2, α) ← A(EK, find)
3. b ←^{R} {0, 1}
4. c ← Enc(m_b)
5. $\tilde{b}$ ← A(c, α, guess)
6. The adversary wins the game if $\tilde{b} = b$.

3.2 Signature Schemes

A signature algorithm can be denoted as S = (Sig-KeyGen, Sig,Ver):

- Sig-KeyGen is an algorithm where input is 1^k, k is a security parameter, and output is (x,y). x is a signing key which is private, while y kept public is a verification key. This operation is denoted as (x,y) ← Sig-KeyGen(1^k).
- Sig is an algorithm where an input is x and a message (m), and an output is a signature (s). This operation is denoted as s←$Sig_{SK}(m)$. If it is clear which signing key is going to be used then this operation can be denoted as: c ←Sig(m).
- Ver is an algorithm where an input is y, a message (m), signature (s) and an output is answer (a). The value of a can be either succeed or fail. This operation is denoted as a←$Ver_y(m, s)$. If it is clear which verification key is going to be used then this operation can be denoted as: a ← Ver(m,s).

Security in Signature Schemes

The main purpose of an attacker is to use forgery. In order to achieve that, the attacker A has to produce a valid signature for the message that he is going to choose. If we suppose that the attacker is in the game then he needs to win the game which is given below:

1. (x,y) ← Sig-KeyGen(1^k)
2. (m, s) ← A(y)

The attacker will be able to succeed if $Ver_y(m, s)$ returns succeed. The capabilities that are given to the attacker are a very important issue. For example, in NMA (No Message Attack) A has an access to y (verification key). In CMA (Chosen Message Attack) the adversary has an access to a signing oracle. Therefore A needs to produce a signature for a new message which he has never used previously.

3.3 Signcryption

The original signcryption scheme was introduced by Zheng in 1997[1]. His purpose was to create a cryptographic system that can not only take benefit of encryption and signature schemes at the same time, but also being computationally efficient.

A signcryption scheme introduced by Zheng can be described in the following way: SC = {Com, Key_s, Key_r, SC, USC}.

- *Com* algorithm generates common parameters shared by all users of the scheme. Hash functions, choice of groups can be given as an example to the common parameters. Input to this algorithm is 'k' and output is 'I. 'I' is a public information. This can be written in the following way:

Choose k

Choose k_q (q|(p-1))

Choose g ϵ Z_p^* of order q

Choose hash function. H:{0,1}* → Z/qZ

Choose hash function. G:{0,1}* →{0,1}^{k_e}

Choose symmetric encryption function E = {Encr, Decr} using k_e

I ← {p,q, g, G, H, E}

- Key_s algorithm executed in the sender's side and is used to generate keys for individual users. These keys are used to unsigncrypt ciphertexts. Input to the algorithm is 'I' and output is (x_s, y_s). It can be wtitten in the following way:

$x_s \overset{R}{\leftarrow}$ Z/qZ

$y_s \leftarrow g^{x_s} \bmod p$

- Key_r algorithm executed in the receiver's side and is used to generate keys for individual users. These keys are used to unsigncrypt ciphertexts. Input to the algorithm is 'I' and output is (x_r, y_r). It can be wtitten in the following way:

$x_r \overset{R}{\leftarrow}$ Z/qZ

$y_r \leftarrow g^{x_r} \bmod p$

- SC is a signcryption algorithm which is used to signcrypt messages. Inputs to the signcryption algorithm are x_s, y_r and m. Output of this algorithm is σ, which is called ciphertext or signcryptext. This can be denoted in the following way:

$n \overset{R}{\leftarrow}$ Z/qZ

k ← $y_r^n \bmod p$

bind ← $y_s \,||\, y_r$

r ← H(m||bind||k)

s ← n / (x_s + r) mod q

K ← G(k)

c ← $Encr_K$(m)

σ ← (c, r, s)

- USC is an unsigncryption algorithm which is used to unsigncrypt ciphertexts. Input to this algorithm is y_s, x_r and σ. Output is m, if ciphertext is valid, otherwise output will be $\perp$. This opearation can be denoted in the following way:

Parse (c, r, s) ← σ

ω ← $(y_s \cdot g^r)^s \bmod p$

k ← $\omega^{x_r} \bmod p$

K ← G(k)

$m \leftarrow Decr_K(c)$

bind $\leftarrow y_s \,||\, y_r$

$r' \leftarrow H(m \,||\, bind \,||\, k)$

If r' =r then output is m

Else output is $\perp$

Note that, k_e and k_q are additional parameters derived from k. As a general rule, the signcrypted message has to be sent from one user to another user. In order to maintain universality and make analysis a bit less complex, users have to establish their own parameters and exchange public keys before communication starts. Furthermore, it is far better to achieve separate set of keys for signed and decrypted messages. In such a way, the possibility of an attack against sender's encryption key or an attack to the confidentiality of the message is eliminated.

It is important to note that, although signcryption is a combination of encryption and signature functions, unfortunately it does not provide all of those functionalities. For example, signature schemes provide authenticity, integrity and non-repudiation. However, it doesn't mean that signcryption schemes will also provide those functionalities. In a signature scheme, anyone can verify the sender's signature by using his/her public key. But in signcryption scheme, the unsigncryption algorithm requires the receiver's private key for validation. Thus in order to define whether or not the given signcryption scheme provides non-repudiation, this scheme must be analyzed separately.

Security in Signcryption

The security issues in signcryption schemes are much more different than encryption and signature schemes. That is because in signcryption we have to take care about insider and outsider attacks. Outsider attacks are those attacks where the attacker knows only public information. But in insider attacks the attacker has access to either to the sender's or reciever's private key. Note that the most important issue during the analyses of security in signcryption is to make sure that confidentiality and authenticity are provided.

Outsider Security:

The main purpose of the attacker is to break either the confidentiality or the authenticity of the signcryption scheme. No matter what the purpose of the attacker is, he/she has access to the sender's signcryption functionality and receiver's unsigncryption functionality. The actions that an attacker needs to take in order to succeed are as followes:

1. $(x_s, y_s) \leftarrow Com(1^k)$
2. $(x_r, y_r) \leftarrow Com(1^k)$
3. $(m_0, m_1, \alpha) \leftarrow A(y_s, y_r, find)$
4. $b \leftarrow^R \{0, 1\}$
5. $u \leftarrow SC(m_b)$
6. $\tilde{b} \leftarrow A(u, \alpha, guess)$

Note that, in order to win the game the attacker needs to have the probability of significance over 0.5. An attacker will succeed if b=$\tilde{b}$. In this scheme an attacker is allowed to send queries to signcryption or de-signcryption oracles. Let's mention that this is almost the same as with encryption schemes. Thus, in the end $m_0 = m_1$.

If in the end of the following steps the probability of significance is greater than zero, then the attacker may achieve forgery.

1. $(x_s, y_s) \leftarrow \mathrm{Com}(1^k)$
2. $(x_r, y_r) \leftarrow \mathrm{Com}(1^k)$
3. $u \leftarrow \mathrm{A}(y_s, y_r)$

The attacker will succeed if u is a valid signcryption of any message that is not queried to the signcryption oracle.

Insider Security:

Insider security is almost similar to the outsider security. The only difference is in the distinguishing of ciphertexts. The attacker has access to the sender's key.

$(m_0, m_1, \alpha) \leftarrow \mathrm{A}(x_s, y_s, y_r, \mathrm{find})$

As long as the attacker has access to x_s, then he/she does not need the signcryption oracle $SC_{x_s,y_r}(\cdot)$. While producing the forgery, the attacker will have access to the receiver's key.

$u \leftarrow \mathrm{A}(y_s, x_r, y_r)$

The attacker does not need to have access to the unsigncryption oracle $USC_{y_s,x_r}(\cdot)$.

It is clear that the insider security provides stronger security than outsider security. In addition, the insider security in confidentiality of a signcryption scheme provides secrecy. Therefore, in insider security past recovery of the messages is difficult. Past recovery of the message is that, if the sender wants to have previously sent messages then he/she needs to store a copy of the message. But some schemes have far better choice. The sender simply may use his/her own private key in order to see the contents of the message from the ciphertext. Those schemes are called past recovery provided schemes. Past recovery was first used in signcryption by Zheng in his own signcryption scheme.

Additionally, we know that in authenticity case, the insider security provides non-repudiation. But unfortunately it is not clear yet whether it can be undesirable in any situations or not.

4 Signcryption primitives

In this chapter we give information about signcryption primitives. Additionally, different types of signcryption schemes will be introduced as well.

4.1 Properties of Signcryption Schemes

In this chapter the properties of signcryption schemes are going to be explained.

4.1.1 Confidentiality and Authenticity

As a general rule, we use signcryption in order to achieve encryption and digital signature, namely confidentiality and authenticity. Confidentiality in signcryption means that, only receiver of a message can read its contents. By this way an attacker can never get any information about the original message. The only thing that the attacker can guess is the length of the message. On the other hand, by authenticity the receiver can verify the sender's identity. By applying authenticity to the message the attacker can never send a message by claiming to be someone else.

4.1.2 Non-repudiation and Signature Verifiability

By applying techniques to ensure non-repudiation to a message, the sender can never deny of having sent the message. The receiver can also easily prove that the message was sent by the sender. It is important to note that, all signature schemes provide non-repudiation so that everyone can verify the signature by the sender's public key. The receiver needs just to forward the message to the third party who can later verify the sender's signature. Actually this is not the main problem in signcryption, because according to the confidentiality only receiver can read the contents of a signcrypted message. But non-repudiation can be achievable by other means too.
There are some kinds of schemes where non-repudiation is not desirable. For example, Krawczyk and Rabin proposed a scheme where only the intended receiver can verify the message's authenticity.
Some signcryption schemes provide non-repudiation in the following way: The receiver restores the message (m) with the signature (s) of that message under some signature scheme (S) and provides (m, s) to the verifier. Then the verifier by using S verifies the sender's signature. This kind of signcryption schemes are called S-verifiable. S-verifiable signcryption schemes are desirable if S is a commonly used signature scheme like DSA.

4.2 Working principles of signcryption schemes

Signcryption is a combination of digital signature and encryption. We can use different kind of digital signature schemes and encryption methods. As an example, implementation of ElGamal's digital signature scheme together with DES or 3DES can be given.

Below we explain working principle of a signcryption scheme based on Elgamal signature scheme with PKE (Public Key Encryption). Here includes encryption and decryption algorithms are also included.

Parameters involved in the signcryption scheme are:

Parameters public to all	p – a large prime number q – a large prime factor of p-1 g – an integer with order q modulo p chosen randomly from [1,…,p-1] Hash – a one-way hash function whose output has at least 128 bits KH – a keyed hash function (E, D) – the encryption and decryption algorithms of a private key cipher
Linda's key	x_a – Linda's private key, chosen at random from [1,…,q-1] y_a – Linda's public key ($y_a = g^{x_a} \mod p$)
John's key	x_b – John's private key, chosen at random from [1,…,q-1] y_b – John's public key ($y_b = g^{x_b} \mod p$)

The following steps are involved in our sample signcryption scheme. Let's note that Linda is a sender of the message and John is a receiver. Linda wants to send a message *m* to John in a secure way.

1. The value x chosen by Linda is in the range of 1,…,q-1
2. Then she uses the hash algorithm in order to hash John's public key and the value x. After hashing Linda will have 128-bit string: K = hash ($y_b^x \mod p$)
3. Then she splits K into two parts, 64-bits in each, k_1 and k_2

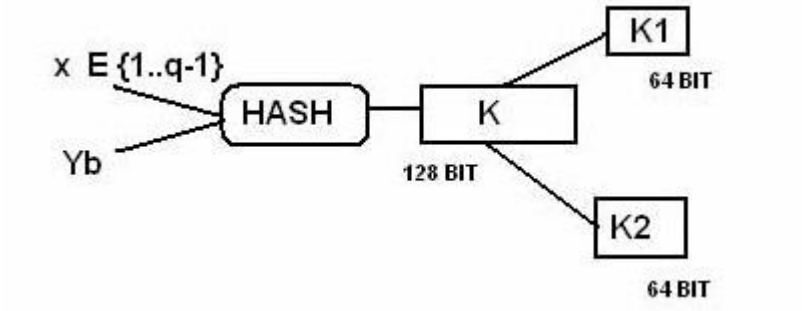

Figure 1: Generation of k_1 and k_2

4. After that, Linda applies PKE to the message *m* with k_1 and sends the signcrypted message to John. In the end of this step she gets a cipher text: c = E_{k_1}(m)

5. In this step she will hash the message using the hash function. Here Linda uses SDSS algorithm and she gets the 128-bit hash: r = H_{k_2}(m)
6. Then she computes s by using her private key (x_a), the value of x, the large value q and the value of r: s = x (r + x_a) mod q

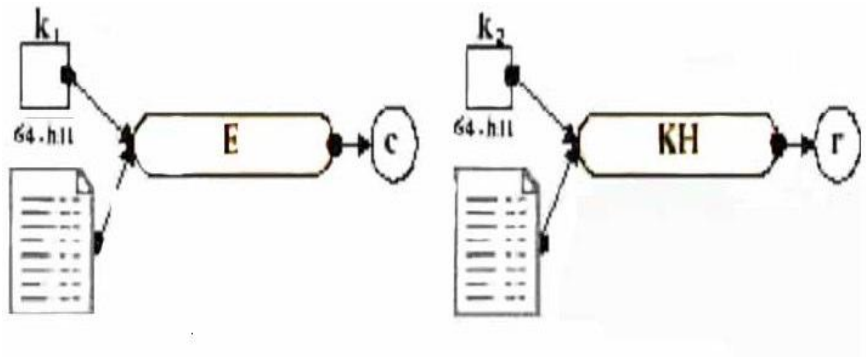

Figure 2: Generation of c and r

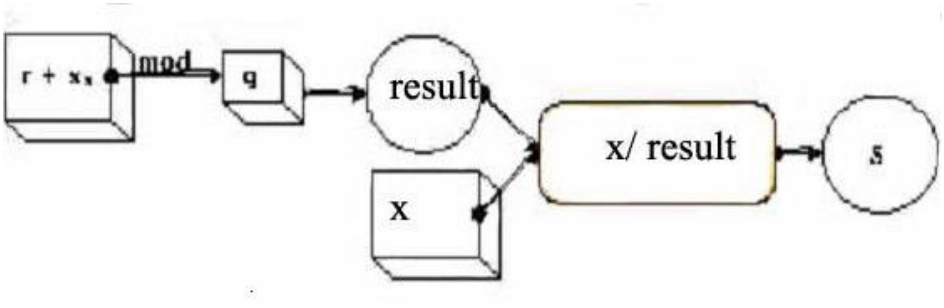

Figure 3: Generation of s

7. Linda has the values of r, c and s. Now it is time to send those values to John. She can do this in two different ways. She can either send all at once or separately. If she sends all separately then she increases the security. Otherwise security of the message is lower. After sending all of those values to John, signcryption in Linda's side is complete.

After completion of signcryption on the Linda's side, now John needs to apply unsigncryption in his side. There are following steps that he needs to apply:

1. John gets 3 values: r, c and s. By using the values of r and s, x_b (his private key), y_a (Linda's public key), p and g he computes a hash value, which is 128-bit long:

   K = hash $((y_a * g^r)^s * x_b \bmod p)$

   The 128-bit long K is then going to be split into 2 parts, 64-bits in each, k_1 and k_2 These values have to be equal to Linda's key values of k_1 and k_2.

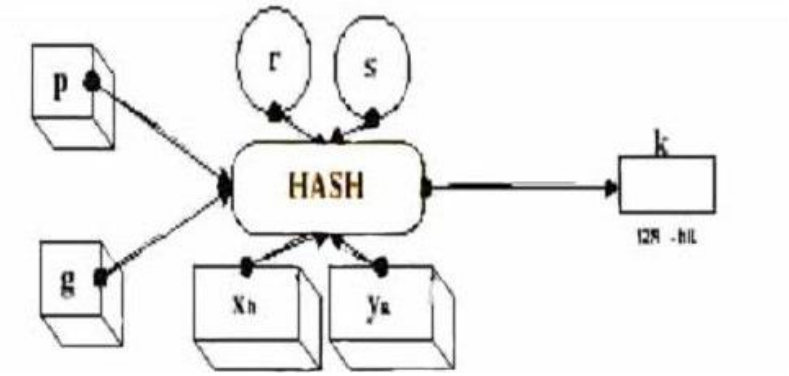

Figure 4: Generating the value of K

2. Then John needs to decipher the c by using k_1. After that John gets the message $m = D_{k_1}(c)$

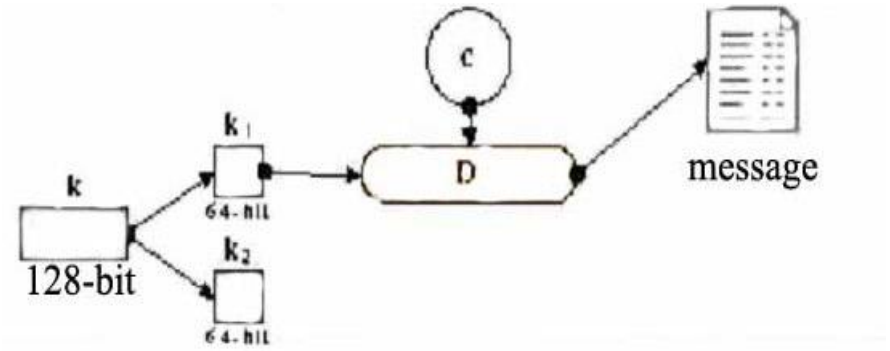

Figure 5: Obtaining message m

3. In the final step John has to apply the hash function on the message m using k_2. Then he compares the value of one-way keyed hash function with the value of r received from Linda. If these 2 values match, then it means that the message m is signed and sent by Linda. Otherwise John knows that the message was either not signed by Linda or it was changed by intruders. In the end, John becomes sure about the originality of the message if following equation is true:

   $H_{k_2}(m) = r$

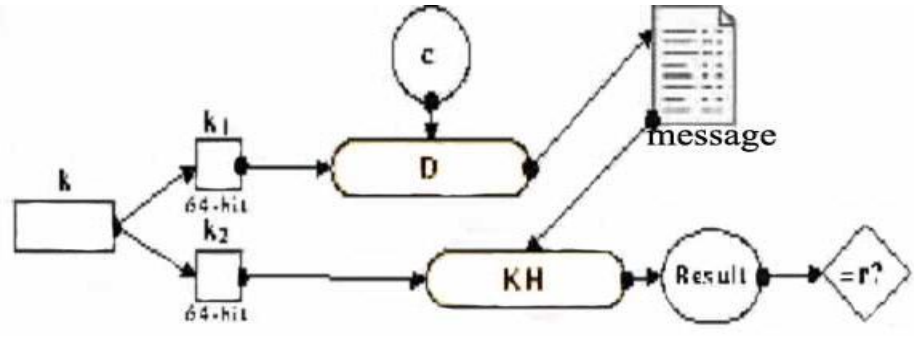

Figure 6: Verification of message m

4.3 Advantages and disadvantages of signcryption

Advantages:

- Low computational cost. Signcryption scheme is an efficient scheme that does two computational steps at once both in signcryption and unsigncryption. If we think to send a single signcrypted message to the receiver, then the cost does not matter so much. But on the other hand, if we think about computational power of the current processors and vass of network traffic then we can be 100% sure about how it is important to take care about computational cost.
- Security. One can argue about the security of signcryption scheme. A simple answer to the proof of security of signcryption schemes is that combination of two complex algorithm would be even more complex than the original algorithms. Complexity of signcryption schemes is described in the picture below:

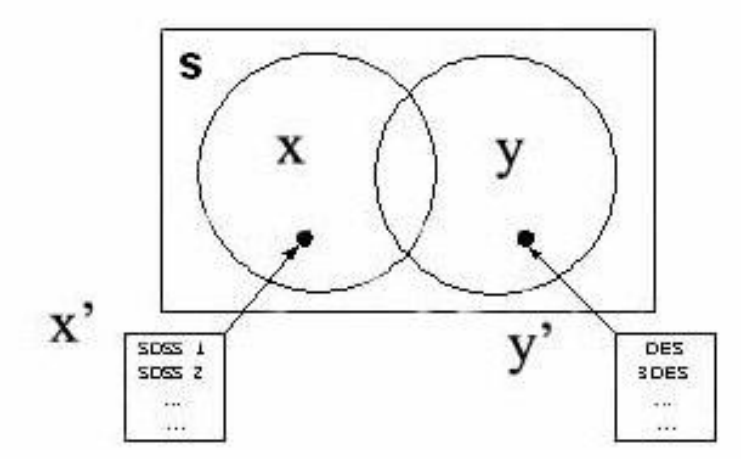

Figure 7: The security of combination of algorithms in signcryption scheme

Disadvantages:

In signcryption schemes the sender has to signcrypt the message using receiver's public key. It has disadvantage when we think about broadcasting of the signcrypted message. Let's assume that a bank needs to send a signcrypted message to a number of share traders. In this situation the bank has to signcrypt the message with each of the share

trader's public key. This is redundant regarding bandwidth consumption and computational resource usage.

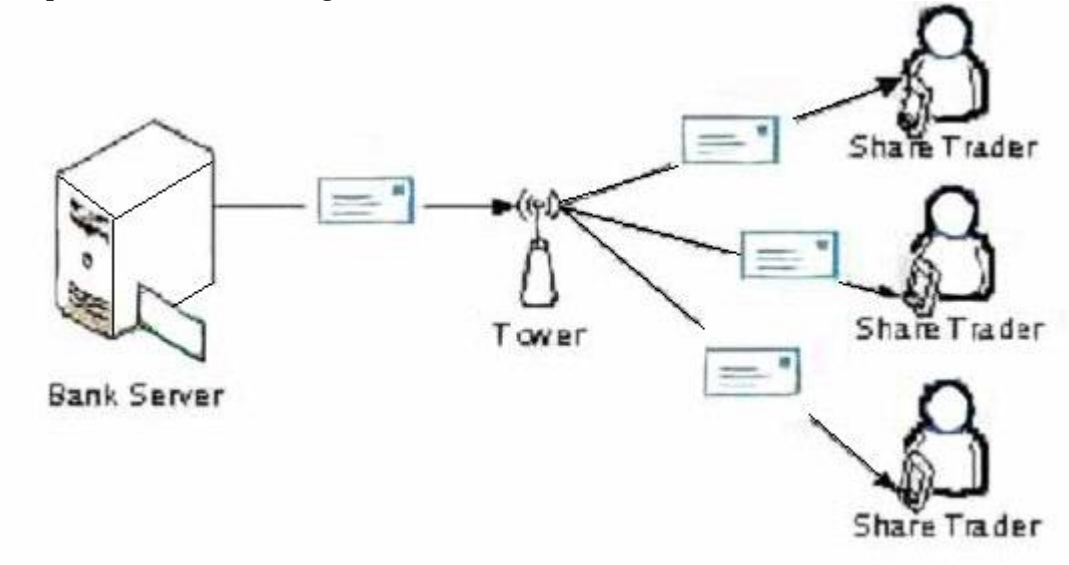

Figure 8: Disadvantage of signcryption schemes

4.4 Types of signcryption schemes

Prior to Zheng's work [15], the usual way to provide both confidentiality and authenticity was to sequentially compose encryption and signature schemes. This can be done in two ways: Encrypt-then-sign and sign-then-encrypt [2]. More information about those schemes can be found in [2]. Below we introduce schemes where both signature and encryption can be integrated. In order to form the signcryption of the message the following basic steps can be followed:

- *Encrypt and Sign*: $\sigma \overset{R}{\leftarrow} Encr(y_r, \mathrm{m}) || Sign(x_s, \mathrm{m})$
- *Encrypt then Sign:* $\sigma \overset{R}{\leftarrow} Encr(y_r, \mathrm{m}) || Sign(x_s, Encr(y_r, \mathrm{m}))$
- *Sign then Encrypt*: $\sigma \overset{R}{\leftarrow} Encr(y_r, \mathrm{m} || Sign(x_s, \mathrm{m}))$

The main difference among those schemes mentioned above is that only 'Encrypt and Sign' works in parallel. It means encryption and signature algorithms don't interact with each other. On the other hand, 'Encrypt then Sign' and 'Sign then Encrypt' don't work in parallel, but serially. It means that, encryption and signature algorithms work in series and input to one algorithm is dependent on the output of the other algorithm. The basic problem in serial schemes is that they are insecure. In 'Encrypt and Sign' scheme an attacker can verify the signature and check whether the signature is equal to m_0 or m_1. Let's guess that Linda wants to send a message to John in 'Encrypt then Sign' scheme. While the message is being sent, Mary intercepts it. Mary removes Linda's signature, puts her own signature and sends to John. In this situation John believes that the message comes from Mary.

In order to make 'Encrypt and Sign' scheme more secure, 'Commit then Encrypt and Sign' scheme was introduced in [9]. 'Commit then Encrypt and Sign' scheme is much

more secure then the others even though Encryption and Signature algorithms work parallel. Linda adds her own public key to the message when she makes computation according to John's public key. Linda adds John's public key if she makes a computation on her own private key. In such a way, the ciphertext may not be forged by any attacker.

Let's assume that Linda wants to send a message to John in 'Sign then Encrypt' scheme. If Linda appends her own information and John's public key to the message, an attacker can't change signature from the message and can no longer deceive John. Precisely, Linda is going to compute the ciphertext in the following way:

$$\sigma' \xleftarrow{R} Encr(y_r, \mathrm{m}||Sign(x_s, \mathrm{m}||y_r)||y_s)$$

Another approach to the signcryption scheme was given by Zheng. In his own signcryption scheme he combined the signature functionality with key encapsulation [1]. In other words, he began from the signature scheme and came to modification of computation of '*k*'. But the modification of '*k*' must be dependent on the receiver's public key. The purpose of doing that is that recovering the value of '*k*' would be possible if and only if anybody knows receiver's private key. In the end, the message can be encrypted by one time symmetric key derived from '*k*'. This approach to signcryption schemes is more efficient than the other approaches.

4.5 SDSS 1 and SDSS 2

Definition of SDSS1. SDSS1 signature scheme consists of 4 algorithms. SC= {Com, Key_s, Sign, Ver}

1. Com(k)
    a. Choose k
    b. Choose k_q (q|(p-1))
    c. Choose g ϵ Z_p^* of order q
    d. Choose H:{0,1}* → Z/qZ
    e. Output: I ←(p, q, g, H)
2. $Key_s(I)$
    a. $x_s \xleftarrow{R}$ Z/qZ
    b. $y_s \leftarrow g^{x_s}$ mod p
    c. Output: (x_s, y_s)
3. Sign(x_s, m)
    a. n $\xleftarrow{R}$ Z/qZ
    b. k ← g^n mod p
    c. r ← H(m || k)
    d. s ← n/(x_s + r) mod q
    e. σ ← (r,s)
    f. Output σ
4. Ver(y_s, m, σ)
    a. Parse (r,s) ← σ
    b. k ← $(y_s \cdot g^r)^s$ mod p
    c. $r' \leftarrow$ H(m || k)
    d. If r' = r then ouput $\top$

Else output $\perp$

Let's mention that if x_s + r ≡ 0 (mod p), then implementation of Sign algorithm will fail although the probability of happening of this case is very small.

Since 1985, ElGamal signature scheme has been studied by many researchers and generalized different forms. More detailed information about different forms of ElGamal digital signature scheme can be found in [3, 10, 18, 19]. Additionally, a shortening of ElGamal based signature been introduced in [20].

In Table 1, two different schemes, SDSS1 and SDSS2, are given which were formed by applying shortening methods to DSS (Digital Signature Stnadard). In Table 1 a shortened versions of DSS, respectively SDSS1 and SDSS2 are given.

Shortened schemes	Signature $(r; s)$ on a message m	Verification of signature	Length of signature
SDSS1	$r = hash(g^x \bmod p, m)$ $s = x/(r + x_a) \bmod q$	$k = (y_a \cdot g^r)^s \bmod p$ check whether $hash(k; m) = r$	$\|hash(\cdot)\|+\|q\|$
SDSS2	$r = hash(g^x \bmod p, m)$ $s = x/(1 + x_a \cdot r) \bmod q$	$k = (g \cdot y_a^r)^s \bmod p$ check whether $hash(k; m) = r$	$\|hash(\cdot)\|+\|q\|$

Table 1: Examples of Shortened and Efficient Signature Schemes

It is important to mention that, SDSS1 is a bit more efficient than SDSS2. SDSS2 includes extra modular multiplication.

Advantages of SDSS1 and SDSS2 over DSS

1. Signature lengths are shorter than DSS. In SDSS1 and SDSS2 signature length is $|hash(\cdot)|+|q|$, while $2|q|$ in DSS.
2. In signature verification there is not required any inversion or division.
3. More securable in random oracle model.

Advantages of signcryption with SDSS1 and SDSS2 are shown in the table below, which was proposed by Zheng:

Various schemes	Computational cost	Computational overhead (in bits)
Signature then encryption based on RSA	EXP=2, HASH=1, ENC=1 (EXP=2, HASH=1, DEC=1)	$\|n_a\| + \|n_b\|$
'Signature then encryption' based on 'DSS + ElGamal encryption'	EXP=3, MUL=1, DIV=1 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=2 ADD=0, HASH=1, DEC=1)	2|q| + |p|
'Signature then encryption' based on 'Schnorr signature + ElGamal encryption'	EXP=3, MUL=1, DIV=0 ADD=1, HASH=1, ENC=1 (EXP=2.17, MUL=1, DIV=0 ADD=0, HASH=1, DEC=1)	|hash(·)| + |q| + |p|

Signcryption SDSS1	EXP=1, MUL=0, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	|KH.(·)| + |q|
Signcryption SDSS2	EXP=1, MUL=1, DIV=1 ADD=1, HASH=2, ENC=1 (EXP=1.17, MUL=2, DIV=0 ADD=0, HASH=2, DEC=1)	|KH.(·)| + |q|

Table 2: Cost of Signcryption and Cost of Signature-Then-Encryption

Comparison of Signcryption with 'Signature then encrypt'

There are three exponentiations that are used for the process of signature-then-encryption and decryption-then-verification in 'signature then encryption' scheme based on Schnorr signature and ElGamal encryption. Two of these three exponentiations are used in calculation of $g^s \cdot y_a^r \bmod p$ or precisely used for verifying Schnorr signature. Using Shamir's technique which is the fastest technique for the calculation of multiple exponential products, $g^s \cdot y_a^r$ can be calculated in (1+3/4) |q| ≈ 1.17 |q| modular multiplications.By using simple 'square and multiply' technique, modular exponentiation can be calculated in 1.5|q| modular multiplications. Hence, by using Shamir and 'square and multiply' techniques the number of modular exponentiations in 'decryption then verification' can be reduced from 3 to 2.17. But unfortunately this reduction can not be applied to sender side modular exponentiations calculation. So in the end the total cost of sender and reciever is 5.17 modular exponentiations.

However the number of modular exponentiations is 1 for signcryption and 2 for unsigncryption in SDSS1 and SDSS2. As long as Shamir's technique can be applied in unsigncryption, then cost of unsigncryption will be 1.17 modular exponentiations. So total cost for computation of signcryption is 1.17 modular exponentiations. Finally,the calculation of the average computational cost is shown in the table below:

Security parameters			Saving average computation cost	Saving in communication overhead
|p|	|q|	|KH(·)|		
512	144	72	58%	70.3%
768	152	80	58%	76.8%
1024	160	80	58%	81.0%
1280	168	88	58%	83.3%
1536	176	88	58%	85.3%
1792	184	96	58%	86.5%
2048	192	96	58%	87.7%
2560	208	104	58%	89.1%
3072	224	112	58%	90.1%
4096	256	128	58%	91.0%
5120	288	144	58%	92.0%
8192	320	160	58%	94.0%
10240	320	160	58%	96.0%

Saving in average computation cost = $\frac{(5.17-2.17)\,modular\ \ exponentiations}{5.17\ modular\ \ exponentiations} = 58\%$

Saving in communication cost = $\frac{|hash(\cdot)|+|q|+|p|-(|KH(\cdot)|+|q|)}{|hash(\cdot)|+|q|+|p|}$

Table 3: Saving of Signcryption over Signature-Then-Encryption Using Schnorr Signature and ElGamal Encryption

Note that, all parameters given in Table 2 and 3 taken from [5].

5 Hybrid conception of signcryption

5.1 Signcryption Tag-KEM

Definition of Signcryption Tag-KEM(SCTK). SCTK consist of 6 algorithms: Com, Key_s, Key_r, Sym, Encap, Decap.

1. **Com** - an algorithm which creates common parameters. Its input is 1^k and output is I (global information for the users of scheme).
2. $\boldsymbol{Key_s}$**-** sender key generation algorithm. Its input is I and output is (x_s, y_s). These key pairs are public/private key pairs which are used to send signcrypted message.
3. $\boldsymbol{Key_r}$**–** receiver key generation algorithm. Its input is I and output is (x_r, y_r). These key pairs are public/private key pairs which are used to receive signcrypted message.
4. **Sym –** Symmetric key generation algorithm. Its input is (x_s, y_r) and output is K(symmetric key) and ω(internal state information).
5. **Encap** – Key encapsulation algorithm. Its input is ω and t (arbitrary tag), while output is E (encapsulation).
6. **Decap** – Decapsulation algorithm. Its input is y_s, x_r and E and t, while output is either K or $\perp$ (error symbol).

Let's note that a simple hybrid signcryption scheme can be formed by combining SCTK and regular DEM. The construction of hybrid signcryption scheme is shown below in Listing 3.

Com(1^k): I $\xleftarrow{R}$ Com(1^k) Return I **Key_s(I):** $(x_s, y_s) \xleftarrow{R} Key_s$(I) Return (x_s, y_s) **Key_r(I):** $(x_r, y_r) \xleftarrow{R} Key_r$(I) Return (x_r, y_r)	**SC(x_S, y_R, m):** (K,ω) $\xleftarrow{R}$ Sym(x_s, y_r) C ← $Enc_K(m)$ E $\xleftarrow{R}$ Encap(ω,C) σ ← (E,C) Return σ **USC(y_S, x_R, σ):** (E,C) ← σ If $\perp$ ← Decap(y_s, x_r, E, C) Return $\perp$ and terminate Else K ← Decap(y_s, x_r, E, C) m ← $Dec_K(C)$ Return m

Listing 3: Construction of a simple hybrid signcryption scheme by combining SCTK and DEM

5.2 Signcryption KEM

The use of hybrid techniques to build signcryption schemes was studied by Dent [13], [14]. By adding authentication to KEM he generalized signcryption KEM.

Definition of Signcryption KEM. Signcryption KEM consists of 6 algorithms: Com, Key_s, Key_r, Encap, Decap, Ver.

1. **Com** - an algorithm which creates common parameters. Its input is k and output is I (global information for the users of the scheme).
2. $\boldsymbol{Key_s}$ **-** sender key generation algorithm. Its input is I and output is (x_s, y_s). These key pairs are public/private key pairs which are used to send signcrypted message.
3. $\boldsymbol{Key_r}$ **-** receiver key generation algorithm. Its input is I and output is (x_r, y_r). These key pairs are public/private key pairs which are used to receive signcrypted message.
4. **Encap** – Key encapsulation algorithm. This algorithm is used to generate an authenticated symmetric key by sender. Its inputs are x_s, y_r and m, while output are K (symmetric key) and e(encapsulation of symmetric key).
5. **Decap** – Decapsulation algorithm. This algorithm is used to get the symmetric key from encapsulation. Its inputs are y_s, x_s and e, while output is K.
6. **Ver** – Verification algorithm. It is used to verify an encapsulation that corresponds to the real sender, receiver and message.Its input is y_s, x_r, m and e, while output is either $\perp$ or $\top$.

Security criteria for signcryption KEM

In this section we give information about security notions, confidentiality and integrity/authentication, of signcryption KEM.

Confidentiality. The confidentiality case in signcryption KEM is very similar to normal KEM. The only difference is that, in signcryption KEM the attacker must be allowed to have access to encapsulation oracle. In order to prove confidentiality of signcryption KEM a game must be played. This game is called IND-CCA2which is played between challenger and attacker: A=(A_1, A_2).

1. The challenger generates global parameters I=Com(1^k), a sender's keypair (x_s, y_s) = Key_s(I), a reciever's keypair (x_r, y_r) = Key_s(I).
2. The attacker runs A_1 by using (y_s, y_r). A_1 terminates by giving state information (*state*). The attacker can query both encapsulation and decapsulation oracle on an input C in this phase. Encapsulation oracle responds to the attackers query by (K,C) = Encap(x_s, y_r) while decapsulation oracle responds by returning K = Decap(y_s, x_r, C).
3. The challenger forms encapsulation (K_0, C^*) by running Encap(x_s, y_r). Additionally, the challenger creates a key K_1 which is at the same length with K_0. Then he/she chooses a bit b ϵ {0,1}. $K^* = K_b$. The challenge's encapsulation is (K^*, C^*).
4. The attacker runs A_2 by using (K^*, C^*) and *state* until he/she guesses correct b' for b. In this phase the attacker can query encapsulation and decapsulation oracles. Decapsulation oracle can not be queried on C^*.

Integrity. In order to accomplish integrity requirement for signcryption KEM the game called LOR-CCA2 must be analyzed.

1. The challenger generates global parameters I=Sim.Com(1^k), a sender's keypair (x_s,y_s) = Com_s(I), a reciever's keypair (x_r,y_r) = Com_s(I). Additionally, the challenger selects a random bit b ϵ {0,1}
2. The attacker runs A on (y_s, y_r). In this phase the attacker can query an encapsulation and a decapsulation oracle. If b=0 the attacker will get response from encapsulation and a decapsulation algorithms in the normal way. Otherwise the answer to the attacker will be formed by the ideal encapsulation and a decapsulation algorithms. A will stop its work when outputting a guess to b' for b.

5.3 Signcryption DEM

Signcryption DEM is almost the same as usual DEM. But they are declared separately, because of having different requirements for data encapsulation mechanism. Signcryption DEM consists of 2 algorithms.

1. **Enc -** Symmetric encryption algorithm. Its inputs are K and m, while output is c.
2. **Dec –** Decryption algorithm. Its inputs are K and c, while output is m.

Security criteria for signcryption DEM

Confidentiality. The confidentiality for a signcryption DEM is the same as normal DEM.

Integrity/Authentication. In order to define integrity of signcryption DEM the game called INT-CCA must be played between an attacker and a challenger.

1. The challenger creates the sequence of the keys: (K_1, K_2, K_3, ...)
2. The attacker runs A. During the execution of A, the attacker can query to the encryption oracle in the format of (i, m) and the answer to the attacker's query will be Enc_{K_i}(m). The same query can be done to decryption oracle and response to this query will be Dec_{K_i}(C). A stops working when a (i^*, C^*) is gained.

The attacker wins the game if $Dec_{K_i}(C^*) \neq \perp$ and C^* was never a response to the oracle query.

By using Signcryption KEM and DEM, the following simple hybrid signcryption scheme can be produced.

1. **SC** – Signcryption algorithm. The input to this algorithm is x_s, y_r and m. Then Encapsulation algorithm runs in order to recover K and e. After that, $Encr_K$(m) will be used to get c. Finally output will be σ = (c,e)
2. **USC** – Unsigncryption algorithm. An input to the algorithm is x_s, y_s and σ. First of all, the algorithm decomposes σ into c and e. Then decapsulation algorithm runs to get K. Proceeding that, decryption algorithm will be used to recover m. Finally, verification algorithm used with the following inputs: y_s, x_r, m and e. If verification algorithm returns ⊤, then output to USC will be m. Otherwise the output will be $\perp$.

5.4 Zheng's signcryption scheme in the hybrid model

In this chapter we give short explaination of Zheng's signcryption scheme in the hybrid security model.

As we saw in previous chapters, Zheng's signcryption scheme unites both signature schemes and hybrid encryption scheme. After giving information about Zheng's signcryption scheme in hybrid model, we can compare the result with the result from other chapters.

Public information: Security parameters k, k_q, k_e **Com(k):** Choose a k-bit prime p Choose a k_q – bit prime q such that q|(p-1) Choose an element g ϵ Z/qZ of order q Choose a cryptographic hash function G:{0,1}* → {0,1}^{k_e} Choose a cryptographic hash function G:{0,1}* → Z/qZ Return I ← (p, q, g, G, H) **Key_S(I):** $x_S \xleftarrow{R}$ Z/qZ $y_S \leftarrow g^{x_S}$ mod p Return (x_r, y_r) **Key_r(I):** $x_r \xleftarrow{R}$ Z/qZ $y_r \leftarrow g^{x_r}$ mod p Return (x_r, y_r)	**Encap(x_s, y_r, m):** n $\xleftarrow{R}$ Z/qZ k ← y_r^n mod p bind ← y_s||y_r r ← H(m||bind||k) s ← n/(x_s + r) mod q K ← G(k) e ← (r,s) Return σ ← (K,e) **Decap(y_s,x_r,e)** Parse (r,s) ← e k ← $(y_s \cdot g^r)^{x_r \cdot s}$ mod p K ← G(k) Return K **Ver(y_s, x_r, m, e):** Parse (r,s) ← e bind ← y_s || x_r k ← $(y_s \cdot g^r)^{x_r \cdot s}$ mod p r' ← H(m||bind||k) If r = r' : Return ⊤ Else Return ⊥

Definition of Zheng's Signcryption KEM. Zheng's signcryption KEM, SC_KEM, consists of the following six algorithms: Com, Key_s, Key_r, Encap, Decap and Ver. Let's note that more detailed information about Zheng's signcryption scheme is given in [4].

Dent states that IND-CCA2, INP-CCA2 and INT-CCA2 are the main security issues in SC_KEM. In IND-CCA2 it is the requirement that the symmetric key output by the Encap algorithm are indistinguishable from a key chosen uniformly at random from K. INP-CCA2 captures the concept that the encapsulation output by Encap should be indistinguishable with respect to the input message m. The notion of INT-CCA2 corresponds to the integrity of encapsulations, in the sense that an adversary can not tamper with or create new encapsulations [27]. According to Dent's results a hybrid signcryption scheme is IND-CCA2 and sUF-CMA secure, whenever SC_KEM and SC_DEM are secure in the respect of their security notions.

Lemma 1. Zheng's signcryption scheme SC is sUF-CMA secure if the SDSS1 signature scheme is sUF-CMA secure.
SC_KEM's being INT-CCA2 secure provides both signature property on the input message and the expected key encapsulation service. This means that SC_DEM needs only to provide confidentiality in the scheme. In other words, SC_DEM does not need to provide authentication service.

From the lemma, we can conclude that SC_KEM is INT-CCA2 secure if SDSS1 signatures are sUF-CMA. Let's note that, accroding to Dent's results the confidentiality of Zheng's signcryption KEM is accomplished by IND-CCA2 and INP-CCA2 notions. More detailed information about IND-CCA2 and INP-CCA2 experiments can be found in [27]

6 Security Models for Signcryption Schemes

In this chapter, we give various models for signcryption schemes, their formal definitions and security issues that have been given in [30,9,6,28].

6.1 The An Model

6.1.1 Privacy in the An model

From the definition of signcryption it is clear that all signcrypton schemes require both sender's secret key and receiver's public key. That is the main difference between signcryption schemes and public key encryption schemes. Because of that, security notions of public key encryption schemes can not be applied to signcryption schemes.
In CPA (Chosen Plaintext Attack) the adversary has access to LR (Left or Right) encryption oracle for the sender's public key. An LR encryption oracle takes (m_0, m_1) as an input and returns encrypted m_d. The main purpose of adversary is to find d which is a hidden bit. The adversary was given access to a decryption oracle for the receiver's public key in order to be able to model security against CCA (Chosen Ciphertext Attack). Let's assume that SC is a signcryption scheme with a security parameter $1^k \in$ N. The input to LR signcryption oracle is (m_0, m_1). The LR signcryption oracle computes σ ← S (x_s, y_s, y_r, m_d) and returns σ back to the adversary. We tag that LR(·). Respectively, the unsigncryption oracle computes m according to the input σ and returns m to the adversary. Let's look at the following example:

$Ex_{\Psi,A}^{ind-attack}\ (1^k)$

d ← {0,1}

I ← C{1^k}

$(x_s, y_s) \leftarrow K_s$(I)

$(x_r, y_r) \leftarrow K_r$(I)

$d' \leftarrow A^{LR(\cdot),O^u(\cdot)}(y_s, y_r)$

If d' = d then return 1, else 0

atk = cpa $\rightarrow O^u(\cdot)$ = U$(y_a, x_b, y_b, \cdot)$

The adversary wins the game if d' = d. The scheme is IND-ATK secure if $\mathrm{Adv}_{SC}^{ind-atk}(1^k, t, q_s, q_u)$ is a negligible function of the security parameter.

6.1.2 Unforgeability in the An Model

In signature schemes the signature generation belongs only to the sender. An attacker in this situation can be anyone. However in signcryption signature generation both sender's key and receiver's key are used. The only one who knows keys corresponding to the public key is the receiver. Because of this distinction, there are two types of definitions in unforgeability: unforgeability by a third person (tuf) and unforgeability by a reciver (ruf). In both cases the attacker has access to a signcryption oracle for the sender's keys [24, 22, 21, 23].

Let's assume that SC is our signcryption scheme and our adversaries are F_p, F_c, G_c and G_p. Adversaries run in two steps: $G_p = (G_{p1}, G_{p2})$ and $G_c = (G_{c1}, G_{c2})$.

$x_{\Psi,F_p}^{tuf-ptxt}(1^k)$ I ← C{1^k} $(x_s, y_s) \leftarrow K_s(\mathrm{I})$ $(x_r, y_r) \leftarrow K_r(\mathrm{I})$ $\sigma \leftarrow F_p^{S(x_s, y_s, y_r, \cdot)}(y_s, y_r)$ $x \leftarrow \mathrm{U}(y_s, x_r, y_r, \sigma)$ If x = ⊥, return 0 If x = m and m was not a query to $S(x_s, y_s, y_r, \cdot)$, return 1, else return 0	$Ex_{\Psi,F_c}^{tuf-ptxt}(1^k)$ I ← C{1^k} $(x_s, y_s) \leftarrow K_s(\mathrm{I})$ $(x_r, y_r) \leftarrow K_r(\mathrm{I})$ $\sigma \leftarrow F_c^{S(x_s, y_s, y_r, \cdot)}(y_s, y_r)$ $x \leftarrow \mathrm{U}(y_s, x_r, y_r, \sigma)$ If x = ⊥, return 0 If σ was not a response from a query to $S(x_s, y_s, y_r, \cdot)$, return 1, else return 0

Listing 4: Experiments 1 and 2

$Ex_{\Psi,G_p}^{ruf-ptxt}(1^k)$ I ← C{1^k} $(x_s, y_s) \leftarrow K_s(\mathrm{I})$ $(x_r, y_r) \leftarrow G_{p_1}(\mathrm{I})$ $(\sigma, x_r', y_r') \leftarrow G_{p2}^{S(\cdot)}(x_r, y_r, y_s)$ $x \leftarrow \mathrm{U}(y_s, x_r', y_r', \sigma)$ If x = ⊥, return 0 If x = m: If $y_r' \neq y_r$, return 0 If m was not a query to $S(x_s, y_s, y_r, \cdot)$, return 1, else return 0	$Ex_{\Psi,G_c}^{ruf-ctxt}(1^k)$ I ← C{1^k} $(x_s, y_s) \leftarrow K_s(\mathrm{I})$ $(x_r, y_r) \leftarrow G_{c_1}(\mathrm{I})$ $(\sigma, x_r', y_r') \leftarrow G_{c2}^{S(\cdot)}(x_r, y_r, y_s)$ $x \leftarrow \mathrm{U}(y_s, x_r', y_r', \sigma)$ If x = ⊥, return 0 If $y_r' \neq y_r$, return 0 If $y_r' = y_r$ and σ was not a response of $E(x_s, y_s, y_r, \cdot)$, return 1, else return 0

Listing 5: Experiments 3 and 4

In experiments 1 and 2, the attack is performed by a third party who has only public information. In experiments 3 and 4 forgery attacks against plaintext and ciphertext respectively are described. The adversary wins if the experiment returns 1.

6.1.3 The An Signcryption Scheme

The composition of encryption and signature are analysed in [25]. The only thing which is not analysed in [25] is DHETM (Diffie Hellman encrypt then MAC). In order to explain DHETM we need to introduce MAC (Message authentication code).

Message authentication code (MAC). MAC consists of three algorithms: KeyGen, T, Ver.

- KeyGen algorithm in input has security parameter (1^k) and outputs k: k ← KeyGen(1^k)
- T is a tagging algorithm. In input this algorithm takes k and m, and outputs $\tau \leftarrow (k, m)$, where $\tau \in \{0,1\}^{Lt}$, Lt ε N
- Ver in input has k, m and $\tau \in \{0,1\}^{Lt}$ and outputs d: d ← Ver(k, m, τ)

Public information: q is a k-bit prime **Com(k):** Choose a k-bit prime p Choose a group G of order q where Choose a generator g of G Return I ← (q, g) $\boldsymbol{Key_a}$**(I),** $\boldsymbol{Key_b}$**(I):** Choose x_s/x_r at random from {1,…,q-1} Compute $y_s = g^{x_s}$ and $y_r = g^{x_r}$ Return $(x_s, y_s)/ (x_r, y_r)$	**SC(x_a, y_a, y_b, m):** Compute k ← H($y_r{}^{x_s}$) Parse k as (k_1, k_2) Compute c ← E_s(k_1,m) Compute τ ← T(k_2,c) Return σ ← (c,τ) **USC(y_a, x_b, y_b, σ):** Parse σ as (c,τ) Compute k ← $H(y_a{}^{x_b})$ Parse k as (k_1, k_2) If Ver(k_2, c, τ) = 0, return ⊤ Compute m ← K(k_1, c) Return m

Listing 6: DHETM

Security analysis in [25] proves that DHETM is IND-CCA and TUF-CTXT secure. The main problem in DHETM is that it doesn't provide RUF-PTXT and RUF-CTXT security. The proper reason is that k ← H($y_r{}^{x_s}$) is being calculated by the sender. However, k ← $H(y_s{}^{x_r})$ can be calculated by the receiver. The other problem in DHETM is non-repudiation. As long DHETM is not RUF-PTXT secure, therefore it does not provide non-repudiation.

6.2 The BSZ Model(Baek, Steinfeld and Zheng)

The BSZ model took the definitions of security for PKE (public key encryption) schemes and signature schemes and adapted them to use in signcryption. The working principle of BSZ signcryption scheme is given in the picture below:

Common Parameter Generation C(1^k):

1. Choose primes p and q such that:
   P has k bits,
   $q|p\text{-}1$ and q > $2^{l_q(1^k)}$ for some function $l_q(\cdot)$
2. Choose an element g of F_p^* of order q
3. Return I = (p,q,g)

Sender/Receiver Key Generation K_s(I)/ K_r(I)

1. Choose x_s/x_r at random from {1,…., q-1}
2. Compute $y_s = g^{x_s}$ mod p/$y_r = g^{x_r}$ mod p
3. Return $(x_s,y_s)/(x_r/y_r)$

Signcyrption S(x_s, y_s, y_r, m)

1. Choose z ← {1,…,q-1}
2. Compute u ← y_r^z mod p
3. Compute k ← G(u)
4. Compute C ← E(k,m)
5. Compute r ← H(m||y_s|| y_r||u)
6. Compute S ← z/(r + x_s) mod q
7. Set σ ← (C,R,S)
8. Return σ

Unsigncryption U(y_s, x_r, y_r, σ)

1. Parse σ as (C,R,S)
2. Compute v ← $(y_s \cdot g^R)^S$ mod p
3. Compute u ← v^{x_r}
4. Compute k ← G(u)
5. Compute m ← D(k,c)
6. If H(m||y_s||y_r||u) = R, return m
7. Return ⊥

Listing 7: The BSZ signcryption Scheme

As it can be seen from Listing 7, (K_s, E, D) symmetric functions have been used in BSZ Model. It also uses two hash functions G: $\{0,1\}^k \rightarrow \{0,1\}^{l_s}$ and H: : $\{0,1\}^* \rightarrow$ {1,.., q-1}

6.2.1 Privacy in the BSZ Model

The confidentiality attack model for signcryption is called FUO (Flexible Unsigncryption Oracle) model. The adversary's main goal is to break confidentiality between sender and receiver. The adversary, Mary, has pk_A, pk_B and access to Linda's both signcryption and unsigncryption oracle. Having access to Linda's signcryption oracle gives an advantage to Mary in a way of exploiting Linda's private key. That is the main difference between FUO attack model for signcryption and other standard attacks.

In the sense of indistinguishability, signcryption schemes are secure of course if there is no any polynomial adversary that can learn more information about the plaintext for the signcrypted text [6].

Definition of FUO-IND-CCA2. Let's assume that SC= (Com, K_r, K_s, SC, USC) and A_c=(A_1, A_2). $A_1$is find-stage algorithm and A_2 is guess-stage algorithm. After making a

number of queries to oracles, the purpose of adversary is to guess the bit b. The only restriction in the queries is that the adversary is not allowed to query C (signcrypted text) to the unsigcryption oracle. It is important to note that the adversary has access to signcryption and unsigncryption oracles.

Now let's look at the experiment given below.

Experiment $Cca2Exp_{SC,A_c}^{fu\text{-}ind-cca2}$(k)

$p_{sc} \leftarrow$ COM(k)

Pick G: $\{0,1\}^* \rightarrow \{0,1\}^l$ at random

Pick H: $\{0,1\}^* \rightarrow$ Z/qZ at random

$(y_r, x_r) \overset{R}{\leftarrow} K_r(\mathrm{k}, cp_{sc})$

$(y_s, x_s) \overset{R}{\leftarrow} K_s(\mathrm{k}, cp_{sc})$

$m_0, m_1, s \leftarrow A_1^{G,H,SC_{x_r,y_s}^{G,H}(\cdot),USC_{x_r}^{G,H}(\cdot)}(k, find, y_r, y_s)$

b $\overset{R}{\leftarrow}$ {0,1}; C $\leftarrow SC_{x_s,y_r}^{G,H}(m_b)$

$b' \leftarrow A_2^{G,H,SC_{x_s,y_r}^{G,H}(\cdot),USC_{x_r}^{G,H}(\cdot)}(k, guess,\ C, y_s, y_r, s)$

if b'=b and C was never queried to $USC_{y_s,x_r}^{G,H}(\cdot)$

return 1

else return 0

It is important that m_0 and m_1 are at the same length, and A_2 never queries unsigncryption oracle with y_a and σ^*. The adversary wins the game if b' = b.

6.2.2 Unforgeability in the BSZ model

Using information that is given in [21], we can define unforgeability for signcryption scheme. Let's assume that the adversary has access to receiver's private and public key. In order to illustrate this attack the experiment where an input is k = |p| ϵ N needs to be made. The formal description of this experiment is given below.

Experiment $ForgeExp_{SC,F}^{cma}$(k)

$cp_{sc} \leftarrow$ COM(k)

Pick G: $\{0,1\}^* \rightarrow \{0,1\}^l$ at random

Pick H: $\{0,1\}^* \rightarrow$ Z/qZ at random

$(y_s, x_s) \overset{R}{\leftarrow} K_s(\mathrm{k}, cp_{sc})$

$(y_r, x_r) \overset{R}{\leftarrow} K_r(\mathrm{k}, cp_{sc})$

If $F^{G,H,SC^{G,H}_{x_A,y_B}(\cdot)}(y_s, y_r, x_r)$ outputs (m,C) such that

1. $USC^{G,H}_{y_s,x_r}(C)$ = m and
2. m was never queried to $SC^{G,H}_{x_s,y_r}(\cdot)$
   return 1
   else return 0

7 Security of Zheng Signcryption Scheme

Broad security analysis of Zheng signcryption scheme has been performed by Baek in [28]. In this chapter we give security information about Zheng signcryption scheme according to his results. In [6] Baek, Steinfeld and Zheng proved that Zheng's SC is secure against GDH problem in FUO-IND-CCA2.

7.1 Confidentiality in Zheng's Signcryption Scheme

In order to prove the confidentiality of Zheng's signcryption scheme some theorems and curtain games will be illustrated. In our first game the attacker tries to attack to Zheng's signcryption scheme in FSO/FUO-IND-CCA. After that game a new game will be introduced by changing current game's rules. Changing the rules of the game means to describe how the computation of variables according to the attacker is being changed. The modification to the games will be continued until achieving the game in a way that the attackers can break the security of symmetric key encryption and a good solution for GDH problem gets found. While making changes in the games, new differences will be found. The probability of those differences can be calculated in the following way.

Lemma 2. Let A_1, A_2, B_1 and B_2 be events defined over some probability space. If Pr[$A_1 \wedge \neg B_1$] = Pr[$A_2 \wedge \neg B_2$], Pr[B_1] ≤ ε and Pr[B_2] ≤ ε then |Pr[A_1]- Pr[A_2]| ≤ ε

Theorem 1. If the GDH assumption holds and the bijevtive one-time symmetric key encryption scheme SKE is PI-SKE secure then Zheng's original signcryption scheme ZSCR is secure in the FSO/FUO-IND-CCA2 sense. Correctly, the following bound holds:

$$InSec_{ZSCR}^{FSO/FUO-IND-CCA2}(t, q_{SC}, q_{USC}, q_G, q_H) \leq 2InSec_{Z_p^*}^{GDH}(t', q_{O^{DDH}}) + InSec_{SKE}^{PI-SKE}(t'') + q_{SC}\left(\frac{q_G+q_H+q_{SC}+q_{USC}+2}{2^{l_q(k)-1}}\right) + \frac{q_H+2q_{USC}}{2^{l_q(k)-1}}$$

where $t' = t + O((q_G)^2 + 1) + O((q_H)^2 + 1) + O(k^3 q_{SC}) + O((k^3+q_G+q_H)\, q_{USC}) + t''(q_{SC} + q_{USC})$ and $q_{O^{DDH}} = (q_{SC} + q_{USC})(q_G+q_H)$.

Game G_0. First of all, oracle generation algorithm of Zheng's signcryption scheme on input parameter, k, will run and achieve cp = (p, q, g, G, H, SKE) where |p|=k, $q>2^{l_q(k)}$, q|(p-1) $Ord_p(g) = q$, G : $\{0,1\}^* \rightarrow \{0,1\}^{l_G(k)}$, H : $\{0,1\}^* \rightarrow Z_q$. SKE is a bijective one-time SKE scheme that consists of encryption and decryption functions. cp and k parameters are going to be used in GK_s and GK_r in order to get sender's and receiver' s public/private key. The sender's public key is y_s^* and private key involves (x_s^*, y_s^*), where $y_s^* = g^{x_s^*}$. While the receiver's public key is y_B^* and private key is (x_r^*, y_r^*) where $y_r^* = g^{x_r^*}$.
In this game the pair of public keys (y_s^*, y_r^*) has to be given to the attacker. When the attacker submits (m_0, m_1), where $|m_0| = |m_1|$ at find stage, ciphertext $C^* = (c^*, r^*, s^*)$ is going to be created according β ϵ {0,1} in the following way:

$c^* = E_{\tau^*}(m_\beta)$;

$r^* = H(m_\beta, y_s^*, y_r^*, K^*)$;

$s^* = x^* / (r^* + x_s^*)$

where $K^* = y_r^{*x^*}$; $\tau^* = G(K^*)$

Note that $x^* \in Z_q^*$. A^{CCA} outputs $\beta' \in \{0,1\}$ at guess stage according to the input C^*.

$$\Pr[S_0] = \frac{1}{2} + \frac{1}{2} Succ_{A^{CCA},ZSCR}^{FSO-IND-CCA2}(k)$$

where S_0 describes the event $\beta' = \beta$. In next games S_i will be used to denote those events.

Game G_1. By changing C^* we achieve $Game_1$. The rules of this game are given below:

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

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

R1-3. Whenever the random oracle H is queried at (m_β, y_s^*, y_r^*, K^*) where $K^* = {y_r^*}^{s^+(r^+ + x_s^*)}$, we respond with r^+.

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

Although some variables being changed in this game, the A^{CCA} has the same distribution in both $Game_0$ and $Game_1$. The only difference is that, in this game $(m_\beta, y_s^*, y_r^*, K^*)$ is queried to H in find stage. The error probability is very small in $Game_1$ and can be defined as followed:

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

Game G_2. In order to create this game some changes were made in the previous game. For example, in G_2 R1-1 and R1-4 rules will be kept, while R1-2 and R1-3 rules going to be canceled. This means that τ^+ and s^+ are used in producing C_+^* . When G and H are being queried by signcryption and unsigncryption oracles, the exact answer is going to be taken from G and H. Let's note that, τ^+ is not being used in other places because of cancelation of rule R1-2. Finally, if $\beta' = \beta$ then A^{CCA} were able to breach the PI-SKE security of bijective one-time symmetric encryption scheme. So we have the following probability:

$$\Pr[S_2] = = \frac{1}{2} + \frac{1}{2} Succ_{A^{PI},SKE}^{PI-SKE}(k)$$

The result is as follows:

$$|\Pr[S_2] - \Pr[S_1]| \leq \Pr[AskKey_2] + \frac{q_{SC} + q_{USC}}{2^{l_q(k)}}$$

Game G_3. In Game G_3 the rule R2-4 needs to be changed, while R2-1, R2-2 and R2-3 remain unchanged, but they will be called R3-1, R3-2 and R3-3 respectively.
In the rule R3-4, which is achieved by modifying the rule R2-4 from the previous game, G and H are going to be replaced by GSim and HSim. Additionally, GList1 and GList2 will be used to simulate random oracle G. GList1 consists of 'input-output' entries for G in the form of (K,τ) and GList2 consists of special 'input-output' entries in the form of $y_R||\omega||(?,\tau)$. Likewise, there are Hlist1 ad HList2 for simulating random oracle H. HList1 consists of simple input-output entries for H in the form of (μ,r), while HList2 have special input-output entries in the form of $y_R||\omega||((m, y_S, y_R,?),r)$. All conditions for GSim and HSim are given below in the picture.

GSim(K)	HSim(m, y_S, y_R, K)
If O(g, ω, y_R, K) = 1 For some $y_R\|\|\ \omega\|\|(?,\tau)$ exists in GList1 Return τ Else $\tau \xleftarrow{R} \{0,1\}^{l_G(k)}$ Return τ Add (K, τ) to GList1	If O(g, ω, y_R, K) = 1 and $y_R\|\|\ \omega\|\|(m, y_S, y_R, ?),r) \in$ HList2 exists in GList1 Return r Else if ((m, y_S, y_R,K),r) exists in HList1 Return r Else $r \xleftarrow{R} Z_q$ Return r Add ((m, y_S, y_R,K), r) to HList1

Listing 8: Random Oracle Simulators GSim and HSim

Note that, GList2 and HList2 are empty throughout this game, because the original signcryption and unsigncryption oracles are being used. Finally we have the following result:

$$\Pr[AskKey_3] = \Pr[AskKey_2]$$

Game G_4.In this game we rename R3-1, R3-2 and R3-3 respectively to R4-1, R4-2 and R4-3. But rule R3-4 needs to altered to R4-4.
In the rule R4-4 replace signcryption oracle by the SCSim (signcryption oracle simulator).

SCSim(y_A^*, (y_R,m))
If $y_R \notin \langle g \rangle \setminus \{1\}$ Return Rej
$\tau \xleftarrow{R} \{0,1\}^{l_G(k)}$; $r \xleftarrow{R} Z_q$; $c \leftarrow E_\tau(m)$; $s \xleftarrow{R} Z_q^*$
If $g^r y_A^* = 1$ Return rej
$\omega \leftarrow (y_A^*\, g^r)^s$
Add $y_R\|\|\ \omega\|\|(?,\tau)$ to GList2
Add $y_R\|\|\ \omega\|\|((m, y_A^*, y_R,?),r)$ to HList2
C ← (c,r,s)
Return C

Listing 9: Signcryption Oracle Simulator

The total probability of outcomes is :

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

In the end, after summing all decryption queries we have the following result:

$|\Pr[AskKey_4] - \Pr[AskKey_3]| \leq q_{SC}(\frac{q_G + q_H + q_{SC} + q_{USC}}{2^{l_q(k)}})$

Game G_5. We rename the rules R4-1, R4-2 and R4-3 to R5-1, R5-2 and R5-3. But after modifying the rule R4-4 we get the rule R5-4. In the rule R5-4 we replace the unsigncryption oracle by a USCSim (Unsigncryption oracle simulator). USCSim can unsigncrypt the unsigncrypted query (yS,C) where C = (c,r,s) witout knowing private key.

USCSim(y_r^*, y_S,C)
If $y_S \notin \langle g \rangle \setminus \{1\}$ Return Rej
Parse C as (c,r,s)
If r $\notin Z_q$ or s $\notin Z_q^*$ or c $\notin SP_c$ Return Reject
ω ← $(y_S\ g^r)^s$
If there exists (K,τ) ϵ GList1 such that O^{DDH}(g, ω, y_r^*, K) = 1 or there exists y_R|| ω′||(?,τ) ϵ GList2 such that O^{DDH}(ω, ω′, y_R, y_r^*) = 1
    m ← D_τ(c)
Else $\tau \xleftarrow{R} \{0,1\}^{l_G(k)}$; Add y_r^*|| ω||(?, τ) to GList2
    m ← D_τ(c)
If there exists ((m, y_S, y_r^* ,K),h) ϵ HList1 such that O^{DDH}(g, ω, y_r^*, K) = 1 or there exists y_R|| ω′||((m, y_S, y_R, ?),h) ϵ HList2 such that O^{DDH}(ω, ω′, y_R, y_r^*) = 1
    If h = r Return m Else Return Reject
Else h $\xleftarrow{R} Z_q$
    Add (y_r^*||w||(m, y_S, y_r^*,?),h) to HList2
    If h = r Return m Else Return Reject

Listing 10: Unsigncryption Oracle Simulator USCSim

Because of DDH oracle, which is used by GSim, HSim and USCSim, the oracles' response is totally dependent on GList and HList. So the view of attacker, A^{CCA}, in $G_4 = G_5$.

$\Pr[AskKey_5] = \Pr[AskKey_4]$

$\Pr[AskKey_5] = Succ^{GDH}_{Z_p^*, A^{GDH}}$ (k)

From 5 games that were given above, we saw that the GDH problem can be solved in $AskKey_i$, where i=2,3,4,5.

7.2 Unforgeability in Zheng's Signcryption Scheme

In this chapter we show formal proof of the assertion that GSL assumption is sufficient for Zheng signcryption scheme in order to achieve unforgeability.

In order to prove that Zheng's signcryption scheme (ZSCR) is unforgeabile we use 'ID reduction technique' introduced in [26]. First of all the attacker makes a passive attack against an identification scheme derived from Zheng's signcryption scheme. Then the attacker will be used to create heavy row that is used to find discrete-logarithm of the Linda's public key.

So our first step is to present IZSCR which is derived from ZSCR. Let's assume that we have the following common parameter:

cp = (k,p,q,g)

which is given to the Prover and the Verifier, where k is a security parameter, |p| = k, q > $2^{l_q(k)}$, q|(p-1), g ϵ Z_p^*. The Prover and the Verifier choose x_p and y_p randomly from Z_q respectively, and compute $y_p=g^{x_p}$ and $y_V=g^{x_V}$ respectively and publish their own key. Then the prover and verifier have to follow the following procedure:

- The prover computes K = y_V^x and sends K to the verifier, where x ϵ Z_q.
- When the verifier receives K from the prover, he sends r to the prover, where r ϵ Z_q.
- The prover sends *s* to the verifier after calculating it : $s = \frac{x}{r+x_p} \epsilon Z_q$.
- The verifier calculates $K' = (y_p g^r)^{sx_V}$. Then he returns 'Accept' if K' = K, otherwise 'Reject'.

Lets note that an attacker in IDSZCR is denoted as A^{IDPSA}.

Now we can show that ZSCR is FiSO-UF-CMA secure by using the lemma given below.

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

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

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

As we mentioned in the beginning of the chapter, our purpose is to construct an attack which tries to find discrete-logarithm of Linda's public key. Therefore we introduce a technique such that the attacker can find discrete-logarithm of Linda's public key.

Heavy Row. Let's assume that A^{IDPSV} is an attacker and he runs an attack against IDZSCR with $Succ_{A^{IDPSV},IDZSCR}^{SI-PSV} \geq \frac{4}{2^{l_q(k)}}$. Additionally, we will take a Boolean matrix F(RA,r) where rows and columns correspond to the attacker's private random strings and the verifier's all possible choices of r (random challenge). Entries in the Boolean matrix are either 0 or 1. If the verifier accepts the attacker's proof then the value of the entry is 1,

otherwise it is 0. If the fraction of 1's in the row is at least $\frac{1}{2}\, Succ^{SI-PSV}_{A^{IDPSV},IDZSCR}(k)$, a row of F(RA,r) will be called *heavy*.

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

Now we need to show that by using the attack with the probability of at least $\frac{4}{2^{l_q(k)}}$, one may easily find the discrete-logarithm of Linda's public key. Let's have a look lemma given below.

Lemma 5. Using the attacker for the scheme IDZSCR, whose success probability is at least (greater than or equal to) $\frac{4}{2^{l_q(k)}}$ and running time is bounded by t_{IDPSV} we construct an attack that finds the discrete-logarithm $x^* \epsilon\ Z_q^*$ of $g^{x^*} \epsilon\ Z_p^*$. Concretely, we obtain the following bound:

$$\mathrm{Succ}^{SI-PSV}_{IDZSCR}(t_{IDPSV}) \leq 2\sqrt{Succ^{DL}_{Z_p^*}(t_{DL})},$$

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

If we combine the results from Lemma 3 and 5 we get the following theorem:

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

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

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

8 Conclusion

8.1 Analysis

The main point of this thesis is security notion of signcryption scheme. While discussing security issues of original Zheng's signcryption scheme, we were able to analyze its security in the hybrid model too. Dent's hybrid model which verifies that Zheng's scheme is insider secure introduced in Chapter 5 . According to the results obtained from Dent's model, it can be said that reductions done in Dent's model makes the model far more strong and secure from the authentication side. But from the confidentiality side, it needs to be stated that a better solution for that model would be worthwhile solution.

Let's note that first it was Dent who stated that signcryption KEM can be used as key agreement mechanisms. According to Dent, even though the signcrytion KEM provides key agrement between two parties, it does not provide authentication. However with outsider security, signcryption KEM can be used to agree a symmetric key with authentication.

It was a strong argument about key encapsulation mechanism in hybrid signcryption in a way that whether encryption KEM can be used as a key agreement mechanism. Definitely encryption KEM allows the users to exchange the message in a secure way. The sender creates a symmetric key and encapsulation of that key by using encryption KEM and the public key of the receiver. The sender sends encapsulation key to the receiver. The receiver can fix the symmetric key by using the decapsulation algorithm.

The main problem with key agreement mechanism is the freshness of the message. For example, John can't make sure whether he is involved in key agreement protocol with Linda or with the attacker. Or the users can not be sure whether the message they got is a replay of the previous one or not.

After inventing signcryption KEM in hybrid model, we can solve some of the suspicions that exist in encryption KEM. By using signcryption KEM Linda (the sender) can be sure that she is in key agreement protocol with John (the receiver). Besides that, she knows also that no one can forge the message and claim being John. The freshness of the message is solved by using nonces and time-stamps.

As we know signcryption scheme in insider security does not provide non-repudiation service. The reason of that is in signcryption scheme it would be difficult, almost impossible, to confirm the ciphertext of a given message for a third party without knowing the receiver's private key. In the case of revealing the receiver’s private key non-repudiation service can also be supplied. But in this situation confidentiality will be lost.

Finally, in the previous chapters we showed that Zheng's cigncryption scheme is FSO/FUO-IND-CCA secure. FSO/FUO-IND-CCA is more securable than IND-CCA although they have many similarities. As we know in FSO/FUO-IND-CCA we allow the attacker to query both signcryption oracle and the unsigncryption oracles. In addition, we showed the unforgeability notions of Zheng's signcryption scheme. By using theorems we showed that, Zheng's signcryption scheme is secure in FiSO-UF-CMA. Note that those security models that we showed in the previous chapters can also be applied to SDSS1 and SDSS2.

8.2 Summary of results

The main point of view in this thesis was signcryption scheme. Throughout the thesis, signcryption scheme were illustrated in different models and their security notions were analyzed.
We began the thesis by giving description of the preliminaries.

Then short information about encryption, signature and signcryption schemes were given in chapter 3.

In chapter 4, we introduced signcryption, signryption properties and primitives, working principles of signcryption, Zheng's signcryption algorithm, advantages and disadvantages signcryption scheme.

In chapter 5 we introduced hybrid blocks of signcryption scheme, signcryption KEM, signcryption DEM and signcryption Tag-KEM. In the end of this chapter the Zheng's signcryption scheme in the hybrid model was illustrated. According to the results of Dent [4], we can state that Zheng's signcryption scheme is much more secure in the hybrid model in the respect of authenticity. But from confidentiality side the better reduction for Zheng signcryption scheme still needed.

In chapter 6, two different models of signcryption were given. The first model in this thesis is An model. The second model proposed by Baek, Steinfeld and Zheng. The results of this model were very strong. The main difference between these two models is that, in the An model the signcryption is related to public key based authenticated encryption. More precisely, An took the ideas about symmetric key setting from [31] and applied them to public key setting. However BSZ model takes the ideas of public key encryption schemes and signature schemes and applies them to the signcryption.

In chapter 7, we introduced security notions of Zheng's signcryption scheme. The main purpose in this chapter was to show confidentiality and unforgeability of Zheng's signcryption scheme in different attacks.

In chapter 8, we gave our final analysis, summary of results and future research.

Nevertheless, there are some open questions that we need to think over. The main question is if we can apply hybrid signcryption scheme to other asymmetric schemes. We should also think about to construct more efficient signcryption schemes with insider security by using KeM-DEM notion.

8.3 Future research

In the thesis we have analyzed the security aspects of signcryption schemes by using ROM. However it would be very interesting to examine the security issues of signcryption outside the ROM.

We would definitely like to see a signcryption scheme providing IND-CCA2 and sUF-CMA insider security, allowing users to choose the key size individually, making parallelization in both the signcryption and designcryption steps.

The other direction for future work should be protocols. Besides providing confidentiality and authenticity, signcryption encompasses identities of the users. This allows us form a key-exchange protocols. On the other hand, signcryption can also be used to reduce the complexity of the protocols.

Bibliography

[1] Yuliang Zheng. Digital signcryption or how to achieve cost (signature & encryption)<< cost (signature) + cost (encryption). In Advances in Cryptolog, CRYPTO '97, volume 1294, pages 165.179. Springer.Verlag, 1997. Unpublished full version (47 pages), dated 1999, available through the author's home page http://www.sis.uncc.edu/ yzheng/papers/signcrypt.pdf.

[2] Clayton D. Smith, Digital Signcryption, Masters Thesis, The University of Waterloo, Canada, 2005.

[3] E. Brickell and K. McCurley. Interactive identifcation and digital signatures. AT&T Technical Journal, pages 73{86, November/Decmber 1991.

[4] Alexander W. Dent. Hybrid cryptography. Cryptology ePrint Archive, Report 2004/210, 2004. http://eprint.iacr.org/2004/210/.

[5] A. Odlyzko. The future of integer factorization. CryptoBytes, 1(2):5{12, 1995. (available at http://www.rsa.com/).

[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] M. Bellare and P. Rogaway: The Game-Playing Technique, International Association for Cryptographic Research (IACR) ePrint Archive: Report 2004/331, 2004.

[8] V. Shoup: Sequences of Games: A Tool for Taming Complexity in Security Proofs , International Association for Cryptographic Research (IACR) ePrint Archive: Report 2004/332,2004.

[9] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Advances in Cryptology . EUROCRYPT 2002, volume 2332, pages 83.107. Springer.Verlag, 2002.

[10] National Institute of Standards and Technology. Digital signature standard (DSS). Federal Information Processing Standards Publication FIPS PUB 186, U.S. Department of Commerce, May 1994.

[11] R. L. Rivest. The MD5 message-digest algorithm. http://www.ietf.org/rfc/-rfc1321.txt.

[12] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In first ACM Conference on Computer and Communications Security, pages 62–73, 1993.

[13] A.W. Dent. Hybrid signcryption schemes with outsider security. In Information Security-ISC 2005, LNCS 3650, pp. 203–217, Springer-Verlag, 2005.

[14] A.W. Dent. Hybrid signcryption schemes with insider security. In Information Security and Privacy-ACISP 2005, LNCS 3574, pp. 253–266, Springer-Verlag, 2005.

[15] Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In B.S. Kaliski Jr., editor, Proc. of Crypto ’97, volume 1294 of LNCS, pages 165–179. Springer-Verlag, 1997.

[16] National Institute of Standards and Technology. Secure Hash Standard, 1995. FIPS Publication 180-1.

[17] R. Canetti, O. Goldreich, and S. Halevi. The random oracle model, revisited. In 30th ACM Symposium on Theory of Computing, pages 209–218. ACM Press, 1998.

[18] K. Nyberg and R. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. Designs, Codes and Cryptography, 7(1/2):61-81, 1996.

[19] C. P. Schnorr. Efficient identifcation and signatures for smart cards. In Advances in Cryptology - CRYPTO'89, volume 435 of Lecture Notes in Computer Science, pages 239{251, Berlin, New York, Tokyo, 1990. Springer-Verlag.

[20] P. Horster, M. Michels, and H. Petersen. Meta-ElGamal signature schemes. In Proceedings of the second ACM Conference on Computer and Communications Security, pages 96{107, New York, November 1994. The Association for Computing Machinery.

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

[22] D. Pointcheval and J. Stern. Security proofs for signature schemes. In Advances in Cryptology - EUROCRYPT ’96, volume 1070 of Lecture Notes in Computer Science, pages 387–398. Springer-Verlag, 1996.

[23] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design validations for discrete logarithm based signature schemes. In Public Key Cryptography

2000, volume 1751 of Lecture Notes in Computer Science, pages 276–292. Springer-Verlag, 2000.

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

[25] J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses. Available at http://eprint.iacr.org/2001/079/, 2001.

[26] K. Ohta and T. Okamoto, On Concrete Security Treatment of Signatures Derived from Identification, Advances in Cryptology - Proceedings of CRYPTO '98, Lecture Notes in Computer Science 1462, pages 354-369,Springer-Verlag, 1998.

[27] Tor E. Bjorstad, Provable Security of Signcryption, Masters Thesis, Norwegian University of Technology and Science, Norway, June 2005.

[28] John Malone-Lee, On the Security of Signature Schemes and Signcryption Schemes, PhD Thesis, Department of Computer Science, University of Bristol, UK, September 2003.

[29] Joonsang Baek, Construction and Formal Security Analysis of Cryptographic Schemes in the Public Key Setting, PhD Thesis, Monash University, Australia, January 2004.

[30] J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses. Available at http://eprint.iacr.org/2001/079/, 2001.

[31] M. Bellare and C. Namprepre, Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm, Advances in Cryptology { Proceedings of ASIACRYPT 2000, Lecture Notes in Com- puter Science 1976, pages 531{545, Springer-Verlag, 2000.

[32] H. Krawczyk, The Order Of Encryption And Authentication For Protecting Communications (Or: How Secure Is SSL?), Advances in Cryptology - Proceedings of CRYPTO 2001, Lecture Notes in Computer Science 2139, pages 310{331, Springer-Verlag, 2001.