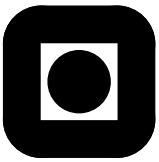
**NTNU**
**Norwegian University of**
**Science and Technology**

**Faculty of Information Technology,**
**Mathematics and Electrical Engineering**



MASTER'S THESIS

BY

STUD.TECHN. Tor Erling Bjørstad

FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS
AND ELECTRICAL ENGINEERING

NTNU

*Date due: June 24, 2005*

## *Discipline: Cryptology*

## *Title: "Provable Security of Signcryption"*

*Purpose of the work: To investigate formal security models and security arguments for signcryption.*

*This Master's Thesis is to be carried out at the Department of Mathematical Sciences under supervision of Professor Alexei Roudakov.*

Trondheim, January 28, 2005.

| | |
|---|---|
| Alexei Roudakov<br>Professor<br>Dept. of Mathematical Sciences | Sverre Smalø<br>Professor<br>Dept. of Mathematical Sciences |

Trond Digernes
Chair
Dept. of Mathematical Sciences

# Abstract

A common goal of cryptographic research is to design protocols that provide a confidential and authenticated transmission channel for messages over an insecure network. The signcryption primitive [Zhe97] provides a common framework for a large class of such protocols. Methods from provable security are often used to establish mathematical guarantees for the security of cryptographic protocols, through formal analysis in a well-defined model of security. After giving a brief introduction to provable security and the signcryption primitive, the main objective of this thesis is a rigorous investigation into the security of Zheng's signcryption scheme under two different security models [BSZ02] [Den04].

# Preface

This Master's Thesis is the result of work done in my final semester, between January and June 2005, of the Master in Technology programme at NTNU. It is intended as a first attempt at independent mathematical research. The thesis is to be read as a general investigation into the security of Zheng's signcryption scheme [Zhe97] in different security models. A main motivation behind the thesis has been to examine existing security models for signcryption, and construct security proofs under these models in a clear and structured manner. This has been of great benefit to my own understanding of the topic, and may hopefully make certain results accessible to a wider audience.

## Structure of the thesis

This thesis is divided into four main parts. The first two chapters are used as an introduction to provable security in general. Chapter 1 gives a brief historical introduction to the material discussed in the remainder of the thesis. In Chapter 2, basic notation and terminology is defined, and several important preliminary concepts are introduced. Simple public-key encryption and signature protocols are defined, together with a general framework for security analysis of such schemes. This includes the Random Oracle Model [BR93] and common game-hopping techniques [Sho04] [BR04] used in security proofs.

The next chapters are used to introduce the main topic of investigation, namely signcryption. In Chapter 3, the signcryption primitive itself is presented, with particular focus on different types of signcryption schemes and common constructions. Chapter 4 discusses common security models for encryption and signatures to the signcryption setting. This covers both the standard models obtained by direct adaptation from the simpler cases of signatures and encryption, as well as a model recently proposed by Dent[Den04] that adapts the KEM/DEM framework used for analysis of hybrid encryption schemes [CS04].

In Chapter 5 and 6, the security of Zheng's signcryption scheme [Zhe97] is analysed with respect to the security goals of of privacy and authenticity. Two different security models are considered. Chapter 5 contains a structured reworking of the results obtained by Baek, Steinfeld and Zheng [BSZ02] with the intention of making their results more accessible. Chapter 6 applies Dent's hybrid model [Den04], and considers how the results of this analysis compare to that of previous models.

Chapter 7 concludes the thesis by discussing alternate and additional security goals that are not covered by in the previous chapters, that may still be of interest. In particular, a modification to Zheng's scheme designed to add the property of forward secrecy [JLLC01]

is given a rigorous examination, and the performance tradeoffs involved in the modification are considered.

## Acknowledgements

Writing is rarely done in a vacuum. I am very grateful to everyone who has been supportive during the writing process, either by giving me technical assistance or simply being sources of inspiration and encouragement.

First of all, I would like to thank some of my fellow master students at NTNU for fruitful discussions, proofreading and technical aid. I am very grateful to Sigurd Segtnan and Vegard Bertelsen for giving me valuable feedback and proofreading early drafts of my thesis, as well as Marius Thaule for assistance with LaTeX. My discussions with Erling Rognerud regarding provable security have also been very fruitful to me, and I can only hope that he had as much benefit from those discussions as myself.

Furthermore, there are several people outside the Department of Mathematical Sciences at NTNU that I am ingratiated to for their kind assistance, feedback and support throughout the previous semester. In particular, Dr. Kristian Gjøsteen at the Department of Telematics at NTNU and Dr. Alex Dent at the Information Security Group at Royal Holloway in London have given me invaluable feedback and shown great patience in the face of my questions. I would also like to thank Prof. Tor Helleseth and Prof. Kjell Jørgen Hole at the Selmer Center at the University of Bergen, for letting me visit their research group for a week in April.

Finally, I would like to thank my supervisor, Prof. Alexei Roudakov. It has been a great pleasure to write this thesis under his guidance, and our discussions and his insightful comments and encouragement have been essential to my understanding of the material.

# Contents

# Listings

# 1

# Introduction

This chapter gives a brief introduction to cryptography, provable security and signcryption, in order to establish a historical context for the remainder of the thesis.

## 1.1 Historical background

Although the history of cryptography can be traced back through the millennia, the codes and ciphers used in antiquity were largely constructed on an ad-hoc basis, and assumed to be secure in the absence of known ways to defeat them. The progression from this state of affairs to the modern approach of building protocols on a solid analytical foundation is often traced to Auguste Kerckhoffs, a Dutchman, who in 1883 formulated six fundamental principles of cipher design in his book *La Cryptographie militaire* [Kah96]. His second principle, often cited as the fundamental assumption of modern cryptography, states that "it should not give any disadvantage if the enciphering mechanism falls into the hands of the enemy". The implication is that the security of a cryptographic scheme should reside in the secret key, rather in the choice of system. This has turned out to be a good starting point for design of cryptographic protocols. Not only does it keep the number of things that need to be kept secret to a minimum, but the secret key is usually the easiest part of a system to change in case of a compromise.

The use of cryptography for civilian purposes is a fairly recent development. Many important historical advances have only been made known to the public in retrospect, after being known but kept classified by military authorities. A rare exception is Claude E. Shannon's groundbreaking article "*Communication Theory of Secrecy Systems*" [Sha49], published in 1949. Shannon applied the machinery of the field he himself created, information theory, to the problem of secret communication. The article introduces concepts such as entropy and redundancy, that have been of great importance for later research. Even more importantly, it marks the final ascension of cryptology from a "black art" to a proper science, with a solid theoretical foundation in mathematics.

The concept of public-key cryptography was introduced to the public in a landmark article by Whitfield Diffie and Martin Hellman in 1976 [DH76], although it had been known by both British and American intelligence for some years. In essence, public-key encryption is Kerckhoffs' principle taken to its logical extreme. Instead of requiring a secret key for encryption, only the key used to decrypt needs to be secret. The central idea behind public-key schemes is that of a mathematical one-way function[1]. For such a function $f$, it is "simple" to compute $f(x)$ given an $x$, while computing $x$ given $f(x)$ is "hard". The idea is that Alice can pick some

[1] Although it is commonly conjectured that one-way functions exist, their existence implies $P \neq NP$ [MVO96].

$x$ as her private key, and publish $f(x)$. A cryptographic operation can then be performed on a message $m$ using $f(x)$, in such a way $x$ is needed to reverse it. Conceptually, this also opens for cryptographic signatures, as Alice can use her knowledge of $x$ to perform a cryptographic operations on a message whose correctness can be verified using $f(x)$. Another important effect of Diffie and Hellman's discovery (together with the public specification of the Data Encryption Standard (DES) in 1975 and the discovery of RSA in 1977) was that it sparked the birth of cryptology as an academic field of research [Sch96].

## 1.2 Provable security

It did not take long for researchers to realize that even though someone who can invert the underlying one-way function trivially can break the security of a public-key scheme, nothing is known about the reverse implication. In fact, naïve attempts at building cryptographic schemes from on well-known assumed one-way problems, such as RSA, are seldom particularly secure. Rabin's cryptosystem [Rab79], published in 1979, was the first public-key scheme shown to be equivalent in security to its underlying mathematical one-way problem. However, it was soon discovered that the security of Rabin's scheme would fall apart if one changed the assumptions of what an attacker might be able to do in an attack.

The concept of provable security was introduced by Goldwasser and Micali in 1982 [GM82]. Their paper introduces the concept of a formal model for the security of a cryptographic protocol. Such a model is divided into two parts; a security model that specifies what it means for a protocol to be "secure", and an adversarial model that specifies what powers an adversary attacking the protocol is allowed to possess. In such a formal setting, one may attempt to present a rigorous analysis of whether breaking the security of the protocol implies breaking an associated one-way function, or another well-known mathematical problem for which no efficient solution exists. In a real sense, the complexity of assumptions is being reduced, from the conjecture that a given cryptographic protocol is secure, to the solvability of a well-studied reference problem.

Provable security has its limitations. If the underlying reference problem turns out to be easy to solve, the security proof of a protocol depending on it fails. Additionally, security proofs give no assurance of security against of adversaries that are not described by the security model used. For instance, a physical implementation of a cryptosystem may leak information to an adversary indirectly, through measurement of side channels such as timing or power consumption [Ber05]. Moreover, if quantum computing becomes feasible in the future, it will invalidate the security of many public-key schemes because it gives the adversary computational powers exceeding that of an universal Turing machine[2]. Finally, there is the question of the proof itself. A security proof requires the same scrutiny and peer review as any other mathematical proof, to detect potential flaws and errors. Many proofs in the literature have been arcane and difficult to grasp, and several examples exist of published results that have not quite held up under scrutiny [SPMLS02] [KM04]. This has given the field of provable security the rather humorous epithet "probable security".

[2]It would also make certain one-way problems, such as factoring large integers, simple to invert.

## 1.3 Signcryption

Encryption and digital signatures have been two of the mainstays of public-key cryptography. As such, it is not surprising that someone would consider ways of combining their features. An obvious approach to doing this is to perform the two operations sequentially, and such constructions have been considered folklore in the cryptographic community for many years. However, such a composition is not particularly efficient. Furthermore, there are various subtle points that must be considered when making such a composition, to ensure that the resulting scheme is as secure as its component parts.

Zheng brought a new perspective to the situation with his 1997 article "*Digital Signcryption or How to Achieve Cost (Signature & Encryption) << Cost (Signature) + Cost (Encryption)*" [Zhe97]. In his article, Zheng asks whether one can perform secure and authenticated message transmission more efficiently than by composition. He proceeds to give a positive answer, by devising a clever scheme that provides both services while saving both computations and bandwidth. The article was also the first to examine the security of such schemes in a holistic manner, considering signcryption as a cryptographic primitive in its own right. In 2002 Baek, Steinfeld and Zheng gave definitive proof that Zheng's proposed scheme is indeed secure, under strong and well-defined notions of security [BSZ02]. Today, signcryption is an active research area with a significant amount of ongoing research, particularly in the "identity-based" setting [LQ04] [DFJW04] [ML04b] [MB04].

# 2

# Preliminaries

This chapter introduces basic notation, definitions and models, in order to construct a fundamental framework for formal security analysis of public-key encryption and signature schemes. It is intended to make the thesis as self-contained as possible, and to serve as a brief introduction to provable security for readers not completely familiar with the subject matter.

## 2.1 Notation and terminology

Clear and consistent notation is a necessity for any piece of scientific writing. It is therefore necessary to define basic notation, terminology, and assumptions that are used throughout this thesis.

The symbols $\mathbb{N}$, $\mathbb{Z}$ and $\mathbb{R}$ are used to represent the natural numbers, the integers and the real numbers, respectively. The set of binary strings of length $n$ is denoted by $\{0,1\}^n$, the set of binary strings of arbitrary length by $\{0,1\}^*$. Given two binary strings $a$ and $b$, their concatenation is denoted by $a||b$. An integer $n$ such that $2^{k-1} \leq n < 2^k$ for some $k \in \mathbb{N}$ is called a *k-bit* integer, and can be represented by a binary string of length $k$.

All groups discussed in this thesis are assumed to be abelian. If $G$ is a group and $g$ is an element of $G$, then $\langle g \rangle$ denotes the cyclic subgroup generated by $g$. For any prime integer $p$, the field of integers modulo $p$ is denoted by $\mathbb{Z}_p$. The cyclic multiplicative group of nonzero elements in $\mathbb{Z}_p$ is denoted $\mathbb{Z}_p^*$.
Given a probability space $X$, the probability of an event $E$ occurring is denoted by $Pr[E]$, and its negation by $Pr[\neg E]$. The conditional probability of an event $E$ occuring given that the event $F$ occurs is denoted $Pr[E|F]$.

An *algorithm* is, in the context of this thesis, defined as a Turing machine that halts after a finite number of steps. The algorithm has access to an input tape (from which input is read), and an output tape (to which output is written). It may also have access to a tape consisting of random bits, in which case it is called probabilistic. Algorithms may return the unique symbols $\perp$ and $\top$ to denote "failure" and "success". Fixed global parameters whose values are used by an algorithm will not be stated explicitly as part of its input, in order to aid legibility.

The notation $x \leftarrow \mathcal{A}(I)$ is used to denote that a deterministic algorithm $\mathcal{A}$ is run with input $I$, and outputs $x$. This notation is also used when $x$ is the result of an explicit computation. If $\mathcal{A}$ is probabilistic, the notation $x \xleftarrow{R} \mathcal{A}(I)$ is used instead. Given a finite set $S$, the notation $x \xleftarrow{R} S$ is used as shorthand for running a probabilistic algorithm that samples $x$ uniformly

at random from $S$.

An *oracle* is a device designed to give an algorithm access to a well-defined computational power. This enables the algorithm to obtain the result of calculations based on knowledge that the algorithm may not access directly, such as the value of a secret key. From the point of view of the algorithm using it, an oracle is a black box that reads queries from an input tape, and writes replies to an output tape. Internally, the oracle is allowed to maintain state information between queries, and may have access to a private random tape. An algorithm $\mathcal{A}$ taking input $I$ and having access to an oracle $O$, is denoted by $\mathcal{A}(I; O)$.
An algorithm is considered *efficient* if it halts after a number of steps that is polynomial in the length of its input. A direct consequence of this is that any efficient algorithm asks a polynomial number of oracle queries. A function $f : \mathbb{N} \to \mathbb{R}$ is considered *negligible* if for every polynomial $p$ there exists $n_0 \in \mathbb{N}$ such that $f(n) \leq \frac{1}{p(n)}$ for all $n \geq n_0$.

A *protocol* is a well-defined series of interactions between two or more parties, designed to accomplish a task. The protocol should both specify algorithms the participants run on their own, and how they should interact with each other. In cryptology protocols, the various protocol participants are commonly antromorphised. Using the convention from [Sch96], when two parties interact in a protocol, they are referred to as Alice and Bob, with an optional third party called Carol. The exception is zero-knowledge protocols, which are analysed as a protocol between a "prover" Peggy and a "verifier" Victor. Additionally, a passive entity eavesdropping on a protocol is commonly known as ("eavesdropping") Eve, while a party actively trying to cheat or subvert the protocol is called ("malicious") Mallory. The term *scheme* is commonly taken to refer to the collection of algorithms the participants run as part of a protocol. The terms scheme and protocol are often conflated, particularly if the protocol interactions themselves are obvious from the context.

## 2.2 Problems and problem solving

Before discussing provable security, it is necessary to have a notion of what a problem is, what it means to solve a problem, and how different problems can be related to each other. This section will give a brief overview of the problems relevant for this thesis. A more thorough discussion of common problems used in cryptography can be found in [MVO96, Chapter 3].

**Definition 2.2.1 (Problem).** A problem $\mathcal{P} = \{I, S, \sigma\}$ consists of a set $I$ containing problem instances, a set $S$ containing of solutions $S$, and a function $\sigma : I \to S$ mapping problem instances to solutions.

As a concrete example, consider the following formulation of the discrete logarithm problem.

**Example 2.2.2 (Discrete Log).** Let $p$ be a $k$-bit prime, and $G$ a cyclic group of order $p$. Furthermore, let $g$ be a generator of $G$ and $\alpha$ an element in $\mathbb{Z}_p$. If $\sigma : g^\alpha \mapsto \alpha$, then $DL_k = \{G^*, \mathbb{Z}_p, \sigma\}$ is the discrete logarithm (DL) problem.

The above example defines a class of problems specified in part by a security parameter $k \in \mathbb{N}$. As $k$ increases, the size of the set of problem instances grows exponentially. Intuitively, $k$ may be viewed as a parameter representing the binary key length corresponding

to a specific problem instance[1]. It is important to note that the hardness of problems such as discrete logarithm are highly dependent on the choice of $G$. Although there exists no efficient algorithm to compute discrete logarithms for an arbitrary group $G$, specific groups and group representations may be easy.

Closely related to the discrete logarithm problem is the Diffie-Hellman problem [DH76]. There are three common variations of this problem, known as the Computational, Decisional and Gap Diffie-Hellman problems.

**Example 2.2.3 (Diffie Hellman).** Let $p$ be a $k$-bit prime, and $G$ a cyclic group of order $p$. Furthermore, let $g$ be a generator of $G$ and $x$, $y$ elements of $\mathbb{Z}_p$. Define the functions $\sigma_{\text{CDH}} : (g^x, g^y) \mapsto g^{xy}$ and $\sigma_{\text{DDH}} : (g^x, g^y, g^z) \mapsto \begin{cases} \top & \text{if } g^z = g^{xy} \\ \bot & \text{if } g^z \neq g^{xy} \end{cases}$. Then $\text{CDH}_k = \{G \times G, G, \sigma_{\text{CDH}}\}$ is the Computational Diffie-Hellman (CDH) problem and $\text{DDH}_k = \{G \times G \times G, \{0,1\}, \sigma_{\text{DDH}}\}$ is the Decisional Diffie-Hellman (DDH) problem.

The Gap Diffie-Hellman problem is fairly new and somewhat less studied, being first proposed in 2001 by Okamoto and Pointcheval [OP01]. It is based on the assumption that there is a computational "difference" between the CDH and the DDH problems: given an oracle that solves the DDH problem, solve the CDH problem. As such, the difference between the CDH and GDH is not in the problem statement as such, but only in the computational powers available. The GDH problem shows up naturally in certain group representations based on elliptic curves, where the DDH problem is known to be easy to solve. The discrete logarithm and Gap Diffie-Hellman problems will be used repeatedly throughout this thesis, together with the "problem" of defeating the security of a cryptographic protocol.

Given a problem formulated in this manner, one may consider the efficiency of an algorithm in solving the problem.

**Definition 2.2.4 (Problem solving).** An algorithm $\mathcal{A}$ solves a problem $\mathcal{P} = \{I, S, \sigma\}$ whenever, if $i \in I$, $s \in S$, $\sigma(i) = s$, then $s$ is also the output of $\mathcal{A}(i)$[2]. The algorithm $\mathcal{A}$ may be deterministic or probabilistic.

**Definition 2.2.5 (Success Rate and Advantage).** The *success rate* of an algorithm $\mathcal{A}$ for solving a problem $\mathcal{P}$ given a running time $t$ and a security parameter $k$, is defined as $Pr\Big[\mathcal{A}(i) = \sigma(i)\Big]$, with the probability taken over all $i$ in the set of problem instances $I$. This is denoted $Succ_{\mathcal{P},\mathcal{A}}(k,t)$.
The *advantage* of $\mathcal{A}$ is defined as $Succ_{\mathcal{P},\mathcal{A}}(k,t) - Succ_{\mathcal{P},\mathcal{R}}(k,1)$, where $\mathcal{R}$ is an algorithm that every time it is run, outputs an answer $s$ picked uniformly at random from $S$. This is denoted $Adv_{\mathcal{P},\mathcal{A}}(k,t)$.
The *advantage function* for $\mathcal{P}$, $Adv_{\mathcal{P}}(k,t)$ is the maximum of $Adv_{\mathcal{P},\mathcal{A}}(k,t)$ taken over all adversaries $\mathcal{A}$ that runs in time $t$.

[1] To be entirely formal, the security parameter should be specified as $1^k$ in unary notation. This follows from the requirement that the running time of an efficient algorithm is bounded by a polynomial in its input length. It is thus necessary to consider running times as polynomials in the security parameter itself, and not in the length of the security parameter (which is $log(k)$ as stated in the main text). Otherwise algorithms that perform elementary operations, such as multiplications modulo $p$, would not be considered efficient. For the remainder of the thesis, this bit of notation is assumed implicitly.

[2] I.e. $s \xleftarrow{R} \mathcal{A}(i)$.

**Remark 2.2.6.** If an algorithm is allowed access to one or more oracles, the number of queries to each oracle also have to be considered as dependent parameters for its success rate and advantage functions. The advantage function for a problem $\mathcal{P}$ is then viewed as the maximum taken over all adversaries with fixed resource usage, with resources including both running time and number of oracle queries.

It is usually only interesting to examine efficient algorithms that have a non-negligible advantage. If no such algorithm can be found, the problem $\mathcal{P}$ is considered to be *hard*.

As part of solving a problem, one might imagine algorithms that transform one problem into a different one. This forms the entire basis for the provable security methodology. If one is able construct an algorithm that transforms the solution of a (hard) problem into the problem of breaking a cryptographic protocol, then one may use an algorithm that has a non-negligible advantage at breaking the protocol to construct one that solves the problem. In absence of a known algorithm that solves the (hard) problem efficiently, one may conclude that the protocol in question is secure.

**Definition 2.2.7 (Reduction).** Given two problems $\mathcal{P} = \{I, S, \sigma\}$ and $\mathcal{P}' = \{I', S', \sigma'\}$ and a pair of algorithms $\mathcal{A}, \mathcal{B}$. Suppose that $\mathcal{A}$ takes an element $i \in I$ as input and outputs one or more elements in $I'$, and that $\mathcal{B}$ takes one or more elements in $S'$ as input and outputs $s \in S$. If $s \leftarrow \mathcal{B}\Big(\sigma'\Big(\mathcal{A}(i)\Big)\Big)$ and $s \leftarrow \sigma(i)$ for all $i \in I$, then the pair $(\mathcal{A}, \mathcal{B})$ is called a (Turing) *reduction* from $\mathcal{P}$ to $\mathcal{P}'$[3]. This is denoted $\mathcal{P} \leq \mathcal{P}'$, and gives a clear indication of the relative hardness of the problems.

For a reduction to be interesting, it is necessary that $\mathcal{A}$ and $\mathcal{B}$ are efficient algorithms. In that case, $\mathcal{A}, \mathcal{B}$ and an algorithm (or oracle) that solves $\mathcal{P}'$ efficiently and with non-negligible probability can be used to construct an algorithm that solves $\mathcal{P}$.

The next lemma shows how the four reference problems defined earlier in this section relate to each other.

**Lemma 2.2.8 (Relation between reference problems).**

$$\text{DDH} \leq \text{GDH} \leq \text{CDH} \leq \text{DL}. \tag{2.1}$$

*Outline of proof.* It is trivial to see that any algorithm $\mathcal{A}$ that computes discrete logarithms can be used to solve the CDH, by using it to compute $x$ from $g^x$ and then calculating $(g^y)^x$, or vice versa. Similarly, any algorithm $\mathcal{B}$ that solves the CDH can be used to solve the DDH by having it compute the value $g^{xy}$ and comparing it to $g^z$. The GDH problem is identical to the CDH problem in its formulation, the sole difference being the additional capability granted by the DDH-oracle. ∎

It is important to note that the status of the *reverse* implications above are known. In particular, it has been conjectured that DDH < CDH = DL, but no proof of this exists.

[3]To simplify the notation, it is assumed that $\sigma'$ is applied to each element output by $\mathcal{A}$

## 2.3 Public-key cryptography

The public-key paradigm is an important tool for the cryptographer. As such, it is essential to know exactly what can be accomplished using public-key methods. This section examines three generic public-key protocols, and discusses their intended design goals.

**DEFINITION 2.3.1 (IDENTIFICATION SCHEME).** A public-key identification scheme **ID** consists of a key generation algorithm $Key_{ID}$ that takes a security parameter $k$ as input, and outputs a private/public keypair $(x, y)$. The identification protocol allows Peggy to prove that she is herself (i.e. knows her private key) to Victor, without giving Victor any new knowledge or enabling Victor to impersonate her. In this paper, protocols based on three-move zero-knowledge protocols are considered [Gol01] [MVO96, Chapter 10.4]. Such a protocol proceeds as follows:

- Commitment: Peggy generates a commitment value $\kappa$ from her private key $x$ and her random tape, and sends it to Victor.
- Challenge: Victor generates a random challenge $r$ and sends it to Peggy.
- Response: Peggy computes an response $s$ using $x$, $\kappa$, $r$ and her random tape, and sends it to Victor.
- Verification: Victor checks whether a specified relation between Peggy's public key $y$ and the values $\kappa$, $r$ and $s$ is valid.

If an identification scheme is secure, then Victor does not learn anything about Peggy's private key from the protocol. Peggy, knowing $x$, should have a negligible chance of failing the verification step of the protocol, while a cheater who only knows $y$ must have a non-negligible chance of failure. Victor may then repeat the protocol several times (or run it several times in parallel), to ensure that the probability of a cheater succeeding becomes arbitrarily low.

**DEFINITION 2.3.2 (SIGNATURE SCHEME).** A public-key signature scheme $\mathbf{S} = \{Key_S, Sign, Ver\}$ consists of three algorithms.

- The key generation algorithm $Key_S$ takes a security parameter $k$ as input, and outputs a private/public keypair $(x_s, y_s)$[4]. Notation: $(x_s, y_s) \xleftarrow{R} Key_s(k)$.
- The signature algorithm $Sign$ takes the sender's private key $x_s$ and a message $m$ as input, and outputs a signature $\sigma$. Notation: $\sigma \xleftarrow{R} Sign(x_s, m)$.
- The verification algorithm $Ver$ takes the sender's public key $y_s$, a message $m$ and a signature $\sigma$ as input, and outputs either $\top$ if the signature is considered valid on the message, or $\bot$ if it is not. Notation: $\{\top, \bot\} \leftarrow Ver(y_s, m, \sigma)$.

[4]The subscript $s$ may be read as a mnemonic for "sender", since it is the key belonging to the person sending a message that is used to sign it. Similarly, the subscript r is used as a mnemonic for the "receiver", since it is the key belonging to the person receiving an encrypted message that is used to encrypt. This notation is used to make the transition to signcryption schemes more clear, since the keys of both sender and receiver are used simultaneously and need to be kept distinct.

The associated protocol is intuitive: Alice signs a message, and sends the message and its signature to Bob.

A signature scheme is designed to protect the authenticity and integrity of a message. When Bob receives a message with Alice's valid signature on it, he should be certain both that the message has not been tampered with by a third party, and that it was Alice who sent it. These are important design goals for cryptographic protocols. Signature schemes also provide non-repudiation. If Alice denies that she signed a given message, Bob is able to give the message and its signature to a neutral judge who can check whether the signature is valid under Alice's key, and thus settle the dispute.

**Definition 2.3.3 (Encryption scheme).** A public-key encryption scheme $\mathbf{E} = \{Key_E, Encr, Decr\}$ consists of three algorithms.

- The key generation algorithm $Key_E$ takes a security parameter $k$ as input, and outputs a private/public keypair $(x_r, y_r)$. Notation: $(x_r, y_r) \xleftarrow{R} Key_E$.
- The encryption algorithm *Encr* takes the intended receiver's public key $y_r$ and a message (*plaintext*) $m$ as input, and outputs a *ciphertext* $c$. Notation: $c \xleftarrow{R} Encr(y_r, m)$.
- The decryption algorithm *Decr* takes the receiver's private key $x_r$ and a ciphertext $c$ as input, and outputs the decryption of that ciphertext, $m$. Notation: $m \leftarrow Decr(x_r, c)$.

The protocol is again directly implied from the specification of the algorithms: Alice encrypts a message, and sends the resulting ciphertext to Bob.

Encryption schemes are intended to provide confidentiality. Neither an eavesdropper Eve nor even an active attacker Mallory should be able to learn anything about the message from the ciphertext, apart from the fact that a message of a certain length is being sent from Alice to Bob.

As public-key encryption is computationally expensive, *hybrid* encryption schemes [CS04] [Den02] are commonly considered to be the best approach for creating efficient public-key encryption schemes. Such schemes use public-key cryptography to transmit a symmetric encryption key, which is used to encrypt the message payload with a symmetric cipher. This yields great efficiency gains, particularly for longer messages. Hybrid encryption has been considered folklore by the cryptographic community for many years, although the first formal security analysis of such constructions was performed by Cramer and Shoup in the late 1990s [CS04].

**Definition 2.3.4 (Hybrid encryption scheme).** A hybrid encryption scheme consists of two parts, a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM).

A key encapsulation mechanism $\mathbf{KEM} = \{Key_K, Encap, Decap\}$ consists of three algorithms.

- The key generation algorithm $Key_K$ takes a security parameter $k$ as input, and outputs a private/public keypair $(x_r, y_r)$. Notation: $(x_r, y_r) \xleftarrow{R} Key_K$.
- The encapsulation algorithm *Encap* takes the intended receiver's public key $y_r$ as input, and outputs a symmetric key $K$ and an encapsulation of that key, $e$. Notation: $(K, e) \xleftarrow{R} Encap(y_r)$.

- The decapsulation algorithm *Decap* takes a the receiver's private key $x_r$ and an encapsulation $e$ as input, and outputs a symmetric key $K$. Notation: $K \leftarrow Decap(x_r, e)$.

The KEM must be sound, in the sense that $Decap(x_r, e)$ should output $K$ whenever $(K, e)$ was been the output of $Encap(y_r)$.

A data encapsulation mechanism **DEM** = $\{Encr, Decr\}$ consists of two algorithms. Both algorithms take as auxiliary input a symmetric key $K$ belonging to a large set $\mathcal{K}$, which is called the *keyspace* of the DEM.

- The encryption algorithm *Encr* takes a key $K$ and a message $m$ as input, and outputs a ciphertext $c$. Notation: $c \leftarrow Encr_K(m)$.
- The decryption algorithm *Decr* takes a key $K$ and a ciphertext $c$ as input, and outputs a message $m$. Notation: $m \leftarrow Decr_K(c)$.

The DEM must be sound, in the sense that $Decr_K\Big(Encr_K(m)\Big) = m$ for any valid key $K$. A security parameter for the DEM is related to the size of $\mathcal{K}$, and will be denoted $k_e \in N$. The keyspace $\mathcal{K}$ is typically the set of bit-strings of length $k_e$.

In a hybrid construction, the security parameter $k$ for the KEM must relate to $k_e$ in such a way that the set of possible symmetric keys output by the KEM is the keyspace of the DEM. If this is the case, a hybrid encryption scheme may be stated in terms of the KEM and DEM. To send an encrypted message to Bob, Alice first runs *Encap* with Bob's public key to generate a symmetric key and an encapsulation. She then to use *Encr* to encrypt her message with the symmetric key just obtained. The encapsulation and ciphertext are sent to Bob, who may use his private key to decapsulate and decrypt in the obvious manner. The relative sizes of the security parameters $k_e$ and $k$ should such that it is no easier for an adversary to attack the DEM directly, than to break the security of the KEM.

The encryption scheme constructed from the KEM and DEM in this manner is provably secure if the KEM and DEM fulfill certain separate security requirements [CS04]. This gives the additional benefit of permitting the KEM and DEM algorithms to be developed and analysed separately.

All key generation, signature and public-key encryption algorithms discussed above are required to be probabilistic. This ensures that unique keypairs are generated whenever the key generation algorithms are run, and foils certain classes of simple eavesdropper attacks against the various schemes[5]. The algorithms used to decrypt ciphertexts and verify signatures are usually required to be deterministic, and this will be assumed throughout the thesis. They are also required to be sound, in the sense that they have negligible probability of returning an incorrect value or failing during a legitimate transaction. In other words, given a valid keypair $(x, y)$ for the appropriate scheme, signatures made with $x$ should verify using $y$, ciphertexts encrypted with $y$ should decrypt to the to the original message using $x$, and encapsulations made with $y$ should decapsulate to the correct symmetric key using $x$.

[5]For example, if Alice encrypts and sends the same message twice, Eve should not be able to detect this from looking at the ciphertexts.

In the schemes defined in this section, certain parameters such as choice of groups used and hash functions are chosen by the key generation algorithms, and published as a necessary part of each user's public key. It is often desirable for several users to use a public-key scheme with some of these parameters in common. In this case, an additional algorithm *Com* is specified, that is run once with $k$ as input and outputs a collection of parameters $I$ that are shared by all the users. If this is done, the key generation algorithm must use the information in $I$ to generate keypairs for each user, which are no longer required to duplicate the values that are contained in $I$. With respect to the other algorithms, $I$ is considered to be a fixed public set of parameters, and will not be stated explicitly as an input parameter.

## 2.4 Security models

Security models for cryptographic schemes are described in terms of an adversary (that is, an efficient algorithm) trying to break the scheme. Such a model provides a formal notion of what it means for the scheme to be secure. The model consists of two main parts: what it means for the scheme to be "broken", and what sort of resources the adversary has access to. Philosophically, it is desirable to make it as simple as possible for the adversary to break the scheme. If the scheme can be shown to be secure (by making a reduction to a hard problem) even when the adversary is powerful and has an easy attack goal, it will also hold up to weaker attacks.

In all public-key systems, a *total break* is defined as a situation where an adversary recovers the private key used in the scheme. Apart from this, what is meant by breaking a cryptographic protocol depends on its purpose. Consider an encryption scheme. An intuitive way to define a "break", is that an adversary is able to recover the plaintext message that has been encrypted. On the other hand, there are situations where partial information may be revealed even though plaintext itself is kept secure. Therefore, another possible definition of a break is that not even a single *bit* of information about the plaintext, other than its length, is revealed to the adversary from the corresponding ciphertext. This is a much stronger test of encryption schemes. Any adversary who can defeat a scheme under the former notion of security, are trivially able to break it under the latter. Yet there may be many classes of adversaries who are able to distinguish between encryptions, yet cannot recover the entire plaintext. The latter notion of what constitutes a break is formalised by the concept of *indistinguishability* (IND) of encryptions. It is the standard notion of security for the confidentiality of encryption schemes. In the case of signature schemes, an immediate notion of a break is that an adversary can forge a signature on a specific message. A stronger test of the scheme is whether the adversary is able to produce a *single* valid signature, on an arbitrary message of his own choice. This is the notion of *existential unforgeability* (UF) of a signature scheme. A signature scheme that is existentially unforgeable ensures that signatures are authentic, that message integrity is provided, and that they are not repudiable.

Similarly, it is necessary to formalize the powers available to an adversary attacking a scheme. Adversaries are modelled as probabilistic algorithms, which means that they are Turing machines that terminate. However, it is often useful to grant it additional capabilities, under a similar philosophy to Kerckhoffs' principle: Faith in the security of a protocol increases

when it is shown to resist powerful attacks[6]. The adversary Eve might conduct a *passive attack* (PA) on an encryption scheme, by attempting to defeat its security by watching transmitted ciphertexts and using public information. On the other hand, the active adversary Mallory might be allowed to decrypt arbitrary ciphertexts of his choice through an oracle, except for the one that he is trying to extract information about. If Mallory is is allowed to do this at any time during an attack simulation, it is called an *adaptive Chosen Ciphertext Attack* (CCA2). Similarly, an adversary trying to make an existential forgery can be access to an oracle that signs arbitrary messages with Alice's key, with the restriction that it needs to forge a "new" signature to be considered successful at its attack. This is called a *Chosen Message Attack* (CMA).

Combining attack goals and capabilities yield different notions of what it means for a scheme to be secure. The canonical security models for analysis of public-key encryption and signature schemes are IND-CCA2 [BDPR98] and UF-CMA [GMR84]. Formal specifications of what it means for a scheme to be secure under these models is given by the following definitions:

**DEFINITION 2.4.1 (IND-CCA2).** The adversary $\mathcal{A}$ is considered to consist of two separate "stages", $\mathcal{A}_1$ and $\mathcal{A}_2$, which are not allowed to communicate directly. When running $\mathcal{A}_1$, it is given Alice's public key as input, and returns two messages of its own choice, as well as any state information it wants to pass on to $\mathcal{A}_2$. One of the messages is then picked at random outside $\mathcal{A}$'s view, and encrypted with Alice's private key. The second stage, $\mathcal{A}_2$, is run with Alice's public key, the state information from $\mathcal{A}_1$, and the encrypted ciphertext as input. It returns a single bit, attempting to guess which of the messages had been encrypted. During the entire simulation, $\mathcal{A}$ is allowed to query a decryption oracle for any ciphertext, with the restriction that it may not decrypt the challenge ciphertext itself. If the advantage of $\mathcal{A}$ at distinguishing between the encrypted messages is negligible in the security parameter $k$, then the scheme is considered to be secure.

**DEFINITION 2.4.2 (UF-CMA).** The adversary $\mathcal{A}$ is given Alice's public key $y$, and attempts to create a message/signature pair $(m, s)$ that is valid under this key. During the simulation, $\mathcal{A}$ is allowed to query a signing oracle with any message, with the restriction that $(m, s)$ is considered an invalid forgery if $m$ was ever queried to the oracle. If the success rate of $\mathcal{A}$ at creating valid forgeries are negligible in the security parameter $k$, then the scheme is considered to be secure.

A brief discussion of UF-CMA is warranted at this point. In 2002, Stern et. al. [SPMLS02] noted that an adversary might be able to create a new signature on a message previously queried to the signing oracle. This would not be allowed under the traditional UF-CMA model, where the adversary by definition does not "break" the scheme if it returns a message previously queried. Nevertheless, producing such a new signature might still be considered a "forgery". Consequently, a stronger notion called *strong* unforgeability (sUF-CMA) has been proposed.

**DEFINITION 2.4.3 (sUF-CMA).** The adversary $\mathcal{A}$ is given Alice's public key $y$, and attempts to create a message/signature pair $(m, s)$ that is valid under this key. During the simulation,

[6] On the other hand, if the adversary is granted too powerful capabilities, it becomes difficult to prove anything at all.

$\mathcal{A}$ is allowed to query a signing oracle with any message, with the restriction that $(m, s)$ is considered an invalid forgery if $m$ was queried to the oracle *and* the oracle replied with $s$.

One may easily construct trivial examples of signature schemes that are UF-CMA, but not sUF-CMA. For instance, consider an UF-CMA-secure scheme where $-s$ is a valid signature on $m$ whenever $s$ is. Philosophically, it is unclear what the adversary "gains" by forging Alice's signature on a message Alice has already signed. In [ADR02], the authors argue that the extra security guarantee sUF-CMA provides is of little practical use, and that regular UF-CMA offer sufficient security guarantees and is easier to work with in practice. An additional problem is that most existing analysis of signature schemes is under the old notion of security. Nevertheless, sUF-CMA is sometimes necessary as a technical part of a security reductions, or as a desirable property in its own right.

Although security models for symmetric encryption schemes in general are outside the scope of this thesis, the above notion of indistinguishability will be used when analysing the data encapsulation mechanisms of hybrid schemes. Adversaries may be given access to an oracle that decrypts chosen ciphertexts, an oracle that encrypts chosen plaintexts, or no oracle at all. The respective security models are referred to as IND-CCA2, IND-CPA and IND-PA. This is somewhat different from the terminology common in analysis of symmetric encryption schemes [BDJR97], but is consistent with terminology used in analysis of signcryption schemes, notably [BSZ02] and [Den04].

## 2.5 Games

In order to instantiate the security notions discussed in the previous section, it is usual to consider *attack games* played between an "experimenter" and the hypothetical adversary algorithm. Games are written in pseudocode, and describe how the experimenter tests the adversary under controlled circumstances as specified by a given security model. Both the experimenter and the adversary are modelled as probabilistic algorithms, which means that each game can be modelled as a probability space, with probabilities taken over all random choices made by either party. That a cryptographic scheme is broken under a given security model can be viewed as an event occuring in this probability space, and the ultimate goal is to bound the probability of this event occuring. To illustrate the game concept, the security models IND-CCA2 and UF-CMA introduced in the previous section are captured by the following games:

---

Public information:
    Encryption scheme $\mathbf{E} = \{Key_E, Encr, Decr\}$.
    Security parameter $k$.

IND–CCA2–experiment:
    $(x_r, y_r) \xleftarrow{R} Key_E(k)$.
    $(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_r; Decr\_Oracle)$.
    $b \xleftarrow{R} \{0, 1\}$.
    $c^* \xleftarrow{R} Encr(y_r, m_b)$.

$b' \xleftarrow{R} \mathcal{A}_2(I, y_r, state, c^*; Decr\_Oracle)$.
$\mathcal{A}$ wins if $b = b'$ and $c^*$ has not been asked to *Decr_Oracle*.

*Decr_Oracle*($c$):
Return $Decr(x_r, c)$.

**Listing 2.1:** IND-CCA2 game for encryption scheme.

Public information:
Signature scheme $\mathbf{S} = \{Key_S, Sign, Ver\}$.
Security parameter $k$.

UF–CMA–experiment:
$(x_s, y_s) \xleftarrow{R} Key_S(k)$.
$(m^*, s^*) \xleftarrow{R} \mathcal{A}(I, y_s; Sign\_Oracle)$.
$\mathcal{A}$ 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 2.2:** UF-CMA game for signature scheme.

The event "$\mathcal{A}$ wins" in these controlled experiments corresponds to the breaking of the cryptographic schemes under the respective notions of security. In slight abuse of notation, the advantage of $\mathcal{A}$ with respect to the problem of breaking the security of a scheme under a specific security model is identified with the advantage of $\mathcal{A}$ at winning the game. To make full security proof, it is necessary to bound this advantage with the probability of another algorithm $\mathcal{B}$ winning a game against some difficult problem. The following game captures the behaviour of an algorithm attempting to solve the Computational Diffie Hellman problem:

Public information:
Security parameter $k$.
Cyclic group $G = \langle g \rangle$, with the order of $G$ equal to a $k$–bit integer.

CDH–experiment:
$x \xleftarrow{R} G$.
$y \xleftarrow{R} G$.
$X \leftarrow g^x$.
$Y \leftarrow g^y$.
$Z \xleftarrow{R} \mathcal{B}(X, Y)$.
$\mathcal{B}$ wins if $Z = g^{xy}$.

**Listing 2.3:** Computational Diffie-Hellman game.

In the above game, $\mathcal{B}$ attempts to solve an instance of the CDH problem. To prove that a specified cryptographic scheme is secure under a given notion of security, the experimenter in that game must be converted into such an adversary $\mathcal{B}$ (against a difficult problem), such that the advantage of $\mathcal{B}$ is connected to that of $\mathcal{A}$. The scheme under consideration is then secure with respect to the problem solved by $\mathcal{B}$, since the specification of $\mathcal{B}$ constitutes an explicit reduction the problem it is solving to the breaking of the scheme.

A complication arising is that if $\mathcal{B}$ is using $\mathcal{A}$ to solve a difficult problem that $\mathcal{B}$ is not able to solve on its own, then $\mathcal{B}$ no longer knows all the parameters of the game it plays with $\mathcal{A}$. For instance, $\mathcal{B}$ may use a supplied problem instance to construct the public key used in the game with $\mathcal{A}$, without knowing the corresponding private key. One may wonder whether the adversary might refuse to play, if it realizes that the game being played has changed from the original. It is therefore important to keep in mind that adversaries are modelled as probabilistic algorithms. The algorithms will accept any well-formed input from the experimenter. However, if a game is altered significantly, the probability that $\mathcal{A}$ wins in the original game but not in the modified one must be kept under control.

In order to transform games in a structured manner, a technique called *game hopping* is applied. The approach has been known to the research community for several years, but has been formalised only recently [Sho04] [BR04]. The underlying idea is to construct a sequence of games, and control the transitions between each game.

**Definition 2.5.1 (Game Hopping).** Let $G_0$ be an initial game, and $E_0$ an event defined in $G_0$. To bound $Pr[E_0]$ (in $G_0$), one may construct a sequence of games $G_1, G_2, \ldots, G_n$ with a corresponding sequence of events $E_1, E_2, \ldots, E_n$ defined in each game. Then

$$Pr[E_0] = \Big(Pr[E_0] - Pr[E_1]\Big) + \Big(Pr[E_1] - Pr[E_2]\Big) + \cdots + \Big(Pr[E_{n-1}] - Pr[E_n]\Big) + Pr[E_n]. \tag{2.2}$$

If each of the transitional probabilities are bounded as well as that of the final game, the goal of bounding $Pr[E_0]$ is attained.

Various standard techniques are available to bound the transitional probabilities between Game $i$ and $i+1$ in the above sum. In particular, the following two lemmas are useful.

**Lemma 2.5.2 (Game hopping lemma).** *Let $G_0$ and $G_1$ be games that are identical until some "failure" event $F$ occurs. Also let $E_0$ and $E_1$ be events defined in the corresponding games, such that $Pr[E_0 \wedge \neg F] = Pr[E_1 \wedge \neg F]$[7]. Then*

$$Pr[E_0] - Pr[E_1] \leq Pr[E_0] - Pr[E_1 \wedge \neg F] \leq Pr[F]. \tag{2.3}$$

*Proof.* The first inequality is trivially true. The second inequality follows from simple manipulation [Sho04]:

$$\begin{aligned} Pr[E_0] - Pr[E_1 \wedge \neg F] &= Pr[E_0 \wedge F] + Pr[E_0 \wedge \neg F] - Pr[E_1 \wedge \neg F] \\ &= Pr[E_0 \wedge F] + Pr[E_1 \wedge \neg F] - Pr[E_1 \wedge \neg F] \\ &= Pr[E_0 \wedge F] \\ &\leq Pr[F] \end{aligned}$$

■

[7] Note that $Pr[F]$ is the same in both games.

The game hopping lemma can be applied whenever a change is made between two games, which causes a specific action by the adversary, such as an oracle query, to make second game proceed differently from the first. This implies that the probability distributions of information sent to the adversary must be identical in both games whenever $F$ has not occurred.

**Lemma 2.5.3 (Distinguisher lemma).** *Let $G_0$ and $G_1$ be two games. Suppose that an experimenter picks $b \xleftarrow{R} \{0,1\}$ at the beginning of each simulation, and proceeds to play $G_b$ with a distinguisher algorithm $\mathcal{D}$ that returns a guess $b'$ of the value of $b$. Then*

$$\left|Pr[b = b'] - \frac{1}{2}\right| = \frac{1}{2}\left|Pr[b' = 0|b = 0] - Pr[b' = 0|b = 1]\right|. \tag{2.4}$$

*Proof.* The result follows from simple manipulation [Den04].

$$\begin{aligned}
\left|Pr[b = b'] - \frac{1}{2}\right| &= \left|Pr[b = b'|b = 0] \cdot Pr[b = 0] + Pr[b = b'|b = 1] \cdot Pr[b = 1] - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|Pr[b' = 0|b = 0] + Pr[b' = 1|b = 1] - 1\right| \\
&= \frac{1}{2}\left|Pr[b' = 0|b = 0] + (1 - Pr[b' = 0|b = 1]) - 1\right| \\
&= \frac{1}{2}\left|Pr[b' = 0|b = 0] - Pr[b' = 0|b = 1]\right|.
\end{aligned}$$

■

The distinguisher lemma can be applied when the underlying probability distributions changes between two games, such as when certain variables sent to the adversary are picked from different distributions. If the change is indistinguishable (by a polynomial-time algorithm), one may construct an algorithm $\mathcal{D}$ that plays either $G_0$ or $G_1$ with the adversary (without knowing which), and outputs 0 whenever $\mathcal{A}$ wins its game. The left hand side of the equation then corresponds to $\mathcal{D}$'s advantage at guessing the hidden bit. The right hand side consists of the probabilities that $\mathcal{A}$ wins $G_0$ and $G_1$, respectively.

## 2.6 Hash functions and the Random Oracle Model

Cryptographic hash functions are very useful in the construction of secure cryptographic protocols. Such hash functions are rather similar to regular hash functions, but have certain additional requirements to prevent information leakage.

**Definition 2.6.1 (Hash function).** A *hash function* is a function $h : X \to Y$ that can be computed efficiently for all $x \in X$. Hash functions are usually expected to provide some degree of compression, mapping elements of a large set $X$ onto a smaller set $Y$[8].
A *collision-resistant hash function* is a hash function satisfying the following three properties:

- One-way: Given a hash value $y \in Y$, it should be infeasible to find a preimage $x \in X$ such that $h(x) = y$.

[8]Commonly, $X$ is the set of bit-strings of arbitrary length up to a fixed maximum $l$, and $Y$ the set of bit-strings of fixed length $m$, with $l >> m$. In the specific case of the well-known cryptographic hash function SHA-1, $l = 2^{63}$ and $m = 160$.

- Weak collision resistance: Given an element $x_1 \in X$, it should be infeasible to find a different element $x_2 \in X$ such that $h(x_1) = h(x_2)$.
- Strong collision resistance: It is infeasible to find two elements $x_1, x_2 \in X$ such that $h(x_1) = h(x_2)$.

A *Message Authentication Code* (MAC) is a family of hash functions $\{h_K\}_{K \in \mathcal{K}}$ for some set $\mathcal{K}$, the keyspace of the MAC. For every $K \in \mathcal{K}$ it is required that there does not exist an efficient adversary $\mathcal{A}$ who, given a collection of input/output-pairs $\Big(x_i, h_K(x_i)\Big)$, can produce a new valid input/output-pair $\Big(x, h_K(x)\Big)$ with $x$ different from each $x_i$. This is known as computation resistance.

Collision-resistant hash functions and MACs are commonly used in cryptographic protocols. However, the above definitions do not capture all the desirable properties one might want in such a function. Their main purpose is to identify a variable-length input value with a shorter, fixed-length hash value in a way that is as random as possible, ideally revealing no information about the input value whatsoever.

Unfortunately, there does not exist any formal definition of a cryptographic hash function that captures *all* the desirable properties a hash function should possess [MVO96]. Indeed, real-world implementations of cryptographic hash functions often fail to behave as randomly as one would hope. The most efficient generic method of finding a collision in a hash function with an $m$-bit output is the so-called *birthday attack*, which runs in $O(2^{m/2})$ operations. Nevertheless, researchers attacking the popular 160-bit cryptographic hash function SHA-1 claim to be able to find collisions in only $O(2^{68})$ operations [WYY05], illustrating the fact that real-world hash functions tend to be less than perfect.

Another problem arising from the lack of a good definition of a cryptographic hash function is that it makes them hard to treat in formal security proofs. While there exist public-key schemes provably secure under the mere assumption that hash functions are collision-resistant [CS98] [CS04], this is the exception rather than the norm. Because of this, non-standard computational models are often employed to analyze idealised hash functions. The *Random Oracle Model* is the most common such model.

**Definition 2.6.2 (Random Oracle Model).** In the Random Oracle Model (ROM), all participants in the scheme being modelled (including adversaries) are given access to one or more *random oracles*. A random oracle $\mathcal{H}$ is an oracle that simulates an idealised hash function $h : X \to Y$. The oracle consists of a (hidden) function $f$, that has been drawn uniformly at random from the set of all functions from $X$ to $Y$. It outputs $f(x)$ whenever an $x \in X$ is received as an input query. In real-world instantiations of a scheme, all random oracles are replaced with regular cryptographic hash function and MACs.

**Remark 2.6.3.** Note that there will be no correlation between $f(x_1)$ and $f(x_2)$ for *any* $x_1 \neq x_2$ in $X$. This is because $f$ is drawn randomly from the set of *all* functions from $X$ to $Y$. This implies that, when simulating a random oracle, one may equivalently use lazy evaluation to simulate the oracle [BR04]. Instead of picking $f$ at random from the set of functions from $X$ to $Y$, the oracle simulator can maintain a list of previous query-response pairs $(x, y)$. Whenever a new $x'$ that is not in the list is queried, a response $y'$ can be drawn uniformly at random

from $Y$, with $(x', y')$ added to the list. This permits analysis of schemes in the random oracle model using the total number of oracle queries as a parameter.

The Random Oracle Model was introduced by Bellare and Rogaway in 1993 [BR93]. The main advantage of using random oracles instead of cryptographic hash functions is that it makes analysis significantly easier. Intuitively, it seems as though any real-world hash function will approximate a random oracle well, since any nontrivial method of distinguishing between a random oracle and the hash function would constitute a de facto attack on the information-hiding properties of the hash function. However, many cryptologists express doubts about how much trust one should place in security proofs made under the Random Oracle Model. Their doubts are reinforced not only by the discovery of weaknesses in existing cryptographic hash functions, but also by the fact that there exist pathological examples of cryptographic schemes that are provably secure under the ROM, but provably *in*secure in *any* real-world instantiation [CGH98] [BBP04]. Therefore, a significant amount of research is going into the development of cryptographic schemes and security models that do not require random oracles.

Despite this, it remains a troublesome fact that very few practical cryptographic schemes exist that are provably secure without using the random oracle assumption. As long as random oracles are required for the majority of efficient and provably secure schemes, they are likely to remain important tools for security proofs. The random oracle assumption also gains strength from the fact that, as argued out in [KM04], the schemes for which the random oracle assumption fails are far removed from practical real-world applicability and established methodologies for secure protocol design.

Another important point regarding the Random Oracle Model is how it supplies a standard method for the construction of signature schemes from identification schemes through what is known as the Fiat-Shamir heuristic [FS87]. As noted in Chapter 2.3, the identification scheme consists of a three-move protocol between Peggy and Victor. To convert it to a signature scheme, Victor is replaced with a random oracle. To sigh a message $m$, Peggy computes a commitment value $\kappa$ using her public key $y$ and her random tape, sends the concatenation $m||\kappa$ to the random oracle to receive a hash value $r$, and computes the same response $s$ as in the identification scheme using $x$, $\kappa$, $r$ and her random tape. At this point, Peggy can publish $\kappa$ and $s$ as her signature on $m$, and any outsider can take the role of Victor and verify that the relation between Peggy's public key $y$, $\kappa$ and $s$ is valid. Alternately, $r$ and $s$ may be published, with the verification being to compute $\kappa$ from $y$, $r$ and $s$ and verifying whether it indeed hashes to $r$.

# 3

# Signcryption

This chapter introduces the signcryption primitive. Different constructions and subtypes of signcryption are discussed. The original scheme proposed by Zheng [Zhe97] is given as an example of efficient signcryption.

## 3.1 The signcryption primitive

The signcryption primitive was introduced by Zheng in 1997 [Zhe97]. Zheng's stated purpose was to create and analyze cryptographic schemes that provide the benefits of encryption and signature schemes simultaneously, while being more computationally efficient than simple composition of encryption and signature. However, some authors also use the term to refer to any scheme that offers this functionality, including direct compositions [ADR02]. In this thesis, the term is used in the more general sense, to admit all schemes that attempts to provide both services.

**Definition 3.1.1 (Signcryption).** A signcryption scheme **SC** = {$Com$, $Key_s$, $Key_r$, $SC$, $USC$} consists of five algorithms.

- The common parameters generation algorithm $Com$ generates parameters shared by all users of **SC**, such as choice of groups and hash functions. It takes the security parameter $k$ as input, and returns public information $I$ used by all the participants. Notation: $I \xleftarrow{R} Com(k)$.
- The sender's key generation algorithm $Key_s$ generates keys for individual users, to be used when signcrypting messages. It takes the public information $I$ as input and returns a private/public keypair $(x_s, y_s)$. Notation: $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
- The receiver's key generation algorithm $Key_r$ generates keys for individual users, to be used when unsigncrypting ciphertexts. It takes the public information $I$ as input and returns a private/public keypair $(x_r, y_r)$. Notation: $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
- The signcryption algorithm $SC$ is used to signcrypt messages. It takes Alice the sender's private key $x_s$, Bob the receiver's public key $y_r$, and a message $m$ as input. It returns a value $\sigma$ which is usually referred to as a ciphertext or signcryptext[1]. Notation: $\sigma \xleftarrow{R} SC(x_s, y_r, m)$.

[1]The term signcryptext is preferred whenever there is danger of confusing the signcrypted ciphertext with a symmetric-key ciphertext used in the same context.

- The unsigncryption algorithm $USC$ is used to unsigncrypt ciphertexts. It takes the sender's public key $y_s$, the intended receiver's private key $x_r$, and a ciphertext $\sigma$ as input. If the ciphertext is valid, it returns a message $m$, otherwise the error symbol $\perp$. Notation: $\{m, \perp\} \leftarrow USC(y_s, x_r, \sigma)$.

The four first algorithms are always probabilistic, the last is usually assumed to be deterministic. To be a sound signcryption scheme, the probability that $USC\Big(y_s, x_r, SC(x_s, y_s, m)\Big)$ does not return $m$ must be negligible for all messages $m$ and valid keypairs $(x_s, y_s), (x_r, y_r)$.

A signcrypted message is being sent from one specific user of the scheme, to another. To maintain full generality and ease of analysis, it is therefore be expected that the users have established common parameters and exchanged public keys before starting communication. Furthermore, it is desirable to maintain a separate set of keys for sending (signing) and receiving (decrypting) messages. In that way, the possibility that a forgery attack against Alice may somehow compromise her encryption key (or vice versa, that an attack on confidentiality reveals information about the signing key used) is ruled out.

One might expect that signcryption, as a combination of encryption and signature functionality, will offer the same four benefits: authenticity, confidentiality, integrity and non-repudiation. However, although signature schemes offers both authenticity, integrity and non-repudiation, signcryption schemes may not. This is due to the fact that, in a signature scheme, anyone can verify Alice's signature on a message by using her public key. In the case of signcryption, the message is hidden, and the unsigncryption algorithm used for validation requires Bob's *private* key as well. Whether a given signcryption scheme offers non-repudiation must therefore be analysed separately from whether confidentiality is provided. Whereas authenticity and confidentiality often are considered to be the primary design goals of signcryption, non-repudiation has received less attention. This is partly due to the fact that non-repudiation may not be desirable in all contexts; while a system designed to provide authenticated and confidential bank transactions must certainly require it, it may not be necessary for session key management in general.

## 3.2 Types of signcryption schemes

The most basic approach to constructing a signcryption scheme, is simply to combine two existing encryption and signature schemes. There are three obvious ways to do this. The following example illustrates how the signcryption itself might be performed.

**EXAMPLE 3.2.1 (NAÏVE COMPOSITIONAL SCHEMES).** Let $Encr$ be a public-key encryption scheme and $Sign$ be a public-key signature scheme. Furthermore, let $\{x_s, y_s\}$ be Alice's signature keypair, and $\{x_r, y_r\}$ Bob's encryption keypair. The following basic compositions can be used to create a signcryption $\sigma$ of a message $m$:

- Encrypt-and-Sign (EaS): $\sigma \xleftarrow{R} Encr(y_r, m) || Sign(x_s, m)$.
- Encrypt-then-Sign (EtS): $\sigma \xleftarrow{R} Encr(y_r, m) || Sign\Big(x_s, Encr(y_r, m)\Big)$.
- Sign-then-Encrypt (StE): $\sigma \xleftarrow{R} Encr\Big(y_r, m || Sign(x_s, m)\Big)$.

These compositions and their associated signcryption schemes are all analysed by An et. al. in [ADR02]. In all three schemes, *Encr* and *Sign* are run once each, and full overhead is incurred both with respect to computational cost and the message expansion caused by *Sign*. The main difference in this respect is that EaS is parallelisible, since the two algorithms do not interact. In contrast, EtS and StE must run *Encr* and *Sign* in series, since the input to one depends on the output from the other. However, none of the schemes are secure as stated. Encrypt-and-Sign does not meet any notion of indistinguishability whatsoever, since any adversary can verify the signature tag and see whether it corresponds to $m_0$ and $m_1$. The problems with EtS and StE are less immediate, and show up in multi-user scenarios. If Alice uses EtS to send a message to Bob and Eve can intercept that message, then Eve can remove the signature and append her own. It will then appear to Bob as though the message came from Eve. On the other hand, if Alice uses StE to send a message to Bob, then Bob can decrypt the message and re-encrypt it with Carol's public key, making it appear to Carol that Alice sent the message to *her*. In [ADR02], a modification of EaS called "Commit then Encrypt and Sign" is proven to be secure while still permitting the parallel execution of *Encr* and *Sign*. The usual heuristic for ensuring multi-user security of public-key schemes is to include information that fixes the sender and/or receiver in the input to the encryption and signature algorithms. Concretely, Alice may append her public key to the message whenever she makes a computation based on Bob's public key, and Bob's public key whenever she makes a computation based on her own private key. In this case, then the resulting ciphertext may not be "dismantled" into its component parts in this manner.

**Example 3.2.2 (Binding sender to receiver).** Consider the case of Alice signcrypting a message to Bob by use of the Sign-then-Encrypt composition presented above. Let $(x_s, y_s)$ be Alice's sending keypair, and $(x_r, y_r)$ Bob's receiving keypair. If Alice appends information about her and Bob's public keys during signcryption, Bob is no longer able to decouple the signature from the encryption and fool Carol into believing that Alice sent the message to *her*. Concretely, if Alice instead computes the ciphertext as $\sigma' \xleftarrow{R} Encr\Big(y_r, m||Sign(x_s, m||y_r)||y_s\Big)$, then Alice's signature is on $m||y_r$, binding the signature to Bob's public key.

A good approach to construct compositional signcryption schemes that are efficient in practice, is to use trapdoor permutations along with a provably secure padding scheme [DFJW04].

A different approach to signcryption is somehow to combine the encryption and signature functionality. However, this author does not know of any secure signcryption schemes that to do this as a single logical operation. The original idea of Zheng [Zhe97] is instead to combine the signature functionality with that of key encapsulation. In short, Zheng starts from a signature scheme created from a three-move identification protocol using the Fiat-Shamir heuristic. His idea is then to modify the initial computation of the "commitment" value $\kappa$ in such a way that it also depends on the receiver's public key. The desired result is that it only should be possible to recover $\kappa$ from the "challenge" $r$ and the response $s$ for someone who knows the receiver's private key. If that is the case, a one-time symmetric key can be derived from $\kappa$ and used to encrypt a message payload. Signcryption schemes resulting from such a construction will be more efficient than those made by composition of a hybrid encryption scheme and a signature scheme, as long as the added cost of modifying the signature scheme is less than that needed to perform encapsulation in the hybrid encryption scheme. For Zheng's scheme, this is indeed the case.

## 3.3 Zheng's signcryption scheme

The signcryption scheme proposed by Zheng is constructed from a signature scheme Zheng refers to as "SDSS1". It is proposed as an example of how to make signatures based on the Digital Signature Standard (DSS) [Sta94] shorter. The main modification is that signing is performed in a large prime order subgroup of $\mathbb{Z}_p^*$, rather than in $\mathbb{Z}_p^*$ itself. Such groups are sometimes referred to as *Schnorr groups*, as they were first suggested for use in cryptography by C. P. Schnorr.

When describing SDSS1 as well as Zheng's scheme, certain notational difficulties arise. Schnorr groups are multiplicative, and can be described (up to isomorphism) by two large primes $p$ and $q$ where $q|(p-1)$, together with an element $g \in \mathbb{Z}_p^*$ of order $q$. However, $\langle g \rangle$ is also isomorphic to the *additive* group $\mathbb{Z}/q\mathbb{Z}$, by the isomorphism $g^x \mapsto x$. Computing this isomorphism in the forward direction is equivalent to solving the discrete logarithm problem. The group $\mathbb{Z}/q\mathbb{Z}$ is again isomorphic to $\mathbb{Z}_q$, and may be given field structure since $q$ is prime. This fact is used explicitly in SDSS1 and Zheng's scheme, which mixes both field operations. The notation $\mathbb{Z}/q\mathbb{Z}$ will be used to represent to the Schnorr group of order $q$, with the implicit understanding that the reverse isomorphism $x \mapsto g^x$ is used to lift group elements into the multiplicative representation when necessary.

**DEFINITION 3.3.1 (SDSS1 SIGNATURE SCHEME).** The SDSS1 signature scheme $\mathcal{S} = \{Com, Key_s, Sign, Ver\}$ consists of four algorithms. Let $k \in \mathbb{N}$ be an overall security parameter, and $k_q \in \mathbb{N}$ an an additional security parameter derived from $k$. The algorithms of SDSS1 may then be specified as follows:

---

$Com(k)$:
Choose a $k-$bit prime $p$.
Choose a $k_q-$bit prime $q$ such that $q|(p-1)$.
Choose an element $g \in \mathbb{Z}_p^*$ of order $q$.
Choose a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}$.
Return $I \leftarrow (p, q, g, \mathcal{H})$.

$Key_s(I)$:
$x_s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$y_s \leftarrow g^{x_s} \mod p$.
Return $(x_s, y_s)$.

$Sign(x_s, m)$:
$n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa \leftarrow g^n \mod p$.
$r \leftarrow \mathcal{H}(m||\kappa)$.
$s \leftarrow n/(x_s + r) \mod q$.
$\sigma \leftarrow (r, s)$.
Return $\sigma$.

$Ver(y_s, m, \sigma)$:
    Parse $(r, s) \leftarrow \sigma$.
    $\kappa \leftarrow (y_s \cdot g^r)^s \mod p$.
    $r' \leftarrow \mathcal{H}(m||\kappa)$.
    If $r' = r$:
        Return $\top$.
    Else:
        Return $\bot$.

**Listing 3.1:** The SDSS1 signature scheme.

The scheme is sound since $(y_s \cdot g^r)^s \equiv (g^{x_s+r})^s \equiv g^n \equiv \kappa \pmod{p}$.

**Remark 3.3.2.** It is important to note that the execution of *Sign* will fail if $x_s + r \equiv 0 \pmod{p}$. However, the probability of this occuring is negligible, since the group order is exponential in $k_q$, and the additive inverse of $x_s$ is unique. In general, there are several "bad" random choices that may be made (such as $x_s = 1$). These are not considered explicitly in the algorithm specification. There are several good reason for this, notably that the probability of either of them occuring is negligible and may be disregarded in practice, and that explicit checking would obscure the presentation of the algorithms.

As commented upon in the previous section, Zheng alters the initial exponentiation with $n$ to be dependent on the receiver's public key $y_r$. Notice that this construction is applicable to many other signature schemes derived from ElGamal, including Schnorr signatures. Zheng's scheme may now be presented in full.

**Definition 3.3.3 (Zheng's signcryption scheme).** Zheng's signcryption scheme $\mathcal{SC} = \{Com, Key_s, Key_r, SC, USC\}$ consists of five algorithms. Let $k \in \mathbb{N}$ be an overall security parameter, and let $k_q, k_e \in \mathbb{N}$ be additional security parameters that are derived from $k$. The algorithms are then specified as follows:

$Com(k)$:
    Choose a $k-$bit prime $p$.
    Choose a $k_q-$bit prime $q$ such that $q|(p-1)$.
    Choose an element $g \in \mathbb{Z}_p^*$ of order $q$.
    Choose a cryptographic hash function $\mathcal{G} : \{0,1\}^* \to \{0,1\}^{k_e}$.
    Choose a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}$.
    Choose a symmetric encryption function $\mathbf{E} = \{Encr, Decr\}$ using
        symmetric keys of length $k_e$.
    Return $I \leftarrow (p, q, g, \mathcal{G}, \mathcal{H}, \mathbf{E})$.

$Key_s(I)$:
    $x_s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
    $y_s \leftarrow g^{x_s} \mod p$.
    Return $(x_s, y_s)$.

$Key_r(I)$:
    $x_r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.

$y_r \leftarrow g^{x_r} \mod p$.
Return $(x_r, y_r)$.

$SC(x_s, y_r, m)$:
$n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa \leftarrow y_r^n \mod p$.
$bind \leftarrow y_s || y_r$.
$r \leftarrow \mathcal{H}(m||bind||\kappa)$.
$s \leftarrow n/(x_s + r) \mod q$.
$K \leftarrow \mathcal{G}(\kappa)$.
$c \leftarrow Encr_K(m)$.
$\sigma \leftarrow (c, r, s)$.
Return $\sigma$.

$USC(y_s, x_r, \sigma)$:
Parse $(c, r, s) \leftarrow \sigma$.
$\omega \leftarrow (y_s \cdot g^r)^s \mod p$.
$\kappa \leftarrow \omega^{x_r} \mod p$.
$K \leftarrow \mathcal{G}(\kappa)$.
$m \leftarrow Decr_K(c)$.
$bind \leftarrow y_s || y_r$.
$r' \leftarrow \mathcal{H}(m||bind||\kappa)$.
If $r' = r$:
Return $m$.
Else:
Return $\perp$.

**Listing 3.2:** Zheng's signcryption scheme.

The scheme is sound since $(y_s \cdot g^r)^{s \cdot x_r} \equiv (g^{x_r})^s \equiv (g^{x_r})^n \equiv \kappa \pmod p$.

**REMARK 3.3.4.** Zheng's signcryption scheme may fail during signcryption for the same reason as SDSS1, namely that $r$ randomly happens to equal the additive inverse of $x_s$. The probability of this happening is negligible. Regarding the relative sizes of the security parameters $k$ and $k_q$, [Zhe97, Table 6] gives suggestions for various values of $k$ based on what is known at the time of writing about the difficulty of computing discrete logarithms in Schnorr groups. For example, for $k = 2048$ the value $k_q = 192$ is suggested. Modern symmetric encryption schemes use keys of length on the order of $128 - 256$ bits, which is thus a suitable range for $k_e$.

When analysing Zheng's scheme, the hash functions $\mathcal{G}$ and $\mathcal{H}$ are modelled as random oracles. In a real world instantiation, [Zhe97] suggests replacing $\mathcal{G}$ with a key derivation function that outputs two keys $(K_1, K_2)$ and $\mathcal{H}$ with a message authentication code $MAC_{K_2}(\cdot)$ ($K_1$ is used with *Encr*). Furthermore, the algorithms are stated for maximum legibility rather than efficiency, and an implementation would make certain obvious optimisations such as computing $\kappa$ directly in *USC*.

Examining the overhead incurred by Zheng's scheme, it becomes apparent that its performance is in fact quite similar to that of SDSS1. Comparing the algorithms *SC* and *Sign*, Alice

must in both cases perform a single exponentiation with the initial random value $x$, run the hash function $\mathcal{H}$ once, and compute $r \leftarrow x/(x_s + r)$. The only additional cost incurred lies in deriving a symmetric key $K$ and using it to encrypt the message. On the receiving end, Bob must in either case exponentiate twice[2] to recover $\kappa$, the only difference being whether the second exponentiation is with $s$ or $s \cdot x_s$. Hence, the additional computational cost incurred by $\mathcal{SC}$ to provide confidentiality is very low compared to the cost of signing a message with SDSS1, as modular exponentiation is by far the most computationally expensive operation performed by either scheme. The message-expansion overhead is also similar for the two schemes, provided that the symmetric encryption scheme does not increase the length of the message. This is because $r$ and $s$ are represented as elements of $\mathbb{Z}/q\mathbb{Z}$ rather than $\mathbb{Z}_p$ in both cases.

Throughout the remainder of this thesis, $\mathcal{SC}$ will be used to refer to Zheng's signcryption scheme of Definition 3.3.2. Furthermore, the algorithms making up SDSS1 and Zheng's scheme will be considered to be public information known by all entities.

[2]However, the cost of computing the two simultaneous modular exponentiations may in practice be lowered to about 1.17 modular exponentiations, using a technique attributed to Shamir [Zhe97, Appendix B].

# 4

# Security models for signcryption

This chapter introduces some of the different security models for signcryption that have been proposed. The models discussed will be used to analyze Zheng's signcryption scheme.

## 4.1 Adaptation of security models

For compositional schemes such as Sign-then-Encrypt, An et. al prove that the resulting signcryption scheme is essentially as secure as the underlying schemes [ADR02]. But when considering signcryption in general, the basic security models introduced for signature and encryption schemes in Chapter 2 can not be used directly. When analysing the confidentiality of signcryption, it is reasonable to maintain the same attack goals as before, namely indistinguishability and existential forgery of signcryptions. If adapted directly, however, an IND-CCA2 adversary will not be able to signcrypt chosen messages from Alice to Bob, since the signcryption algorithm requires Alice's private sending key as input. Hence, for signcryption, an IND-CCA2-adversary should be given access to *two* oracles; one for signcryption and one for unsigncryption. This may be expressed through the following IND-CCA2 attack game:

---

Public information:
Signcryption scheme $\mathbf{SC} = \{Com, Key_s, Key_r, SC, USC\}$.
Security parameter $k$.

IND–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$.
$b \xleftarrow{R} \{0, 1\}$.
$\sigma^* \xleftarrow{R} SC(x_s, y_r, m_b)$.
$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$.
$\mathcal{A}$ wins if $b = b'$ and $\sigma^*$ has not been asked to *USC_Oracle*.

*SC_Oracle*$(m)$:
Return $SC(x_s, y_r, m)$.

*USC_Oracle*$(\sigma)$:

Return $USC(y_s, x_r, \sigma)$.

**Listing 4.1:** IND-CCA2 game for signcryption.

In [BSZ02], Baek, Steinfeld and Zheng give the first full-fledged proof of security of Zheng's signcryption scheme. They do this under a stronger security model for confidentiality, which they call the flexible unsigncryption oracle (FUO) model. In the FUO-IND-CCA2 security model, the unsigncryption oracle given to the adversary takes an *arbitrary* sender's public key $y_s'$ as input, in addition to the ciphertext $\sigma$ to be unsigncrypted. Hence, the adversary is allowed to unsigncrypt messages from anyone to Bob, rather than just from Alice. The idea is that this may be a more realistic assumption in a multi-user setting, and that the additional capability granted may aid the adversary in distinguishing between ciphertexts.

For adversaries against the authenticity and integrity of signcryption, it is necessary to consider two different cases, insider and outsider attacks. In an outsider attack, Mallory attempts to forge a signcryptext from Alice to Bob. An insider attack occurs when Mallory attempts to forge signcryptexts from Alice to himself. The latter may be easier for Mallory, since he then knows the private receiving key $x_r$. Whether a signcryption scheme is required to be insider secure depends on whether legitimate users of the scheme are trusted in the intended application of the scheme. This thesis will only consider the stronger insider-secure attack model. In that case, the CMA oracle is adapted as a signcryption oracle sending messages from Alice to Mallory. The resulting UF-CMA security model is expressed by the following game:

Public information:
Signcryption scheme $\mathbf{SC} = \{Com, Key_s, Key_r, SC, USC\}$.
Security parameter $k$.

UF–CMA–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$(m^*, \sigma^*) \xleftarrow{R} \mathcal{A}(I, y_s, x_r; SC\_Oracle)$.
$\mathcal{A}$ wins if $m^* \leftarrow USC(y_s, x_r, \sigma^*)$ and $m^*$ was never queried to $SC\_Oracle$.

$SC\_Oracle(m)$:
Return $SC(x_s, y_r, m)$.

**Listing 4.2:** UF-CMA game for signcryption.

The notion of strong unforgeability of signcryptions (sUF-CMA) relates to the above unforgeability notion as previously. For a signcryption scheme to be sUF-CMA secure, the the adversary is not even able to produce a new signcryption of a message previously queried to the signcryption oracle.

## 4.2 Hybrid signcryption

In [Den04, Section 5] [Den05], Dent proposes a general model for hybrid signcryption schemes with insider security. He then proceeds to prove that such a hybrid scheme is secure whenever the underlying signcryption KEM and DEM is secure. This section will be used to summarise his construction, security models and main results.

**DEFINITION 4.2.1 (SIGNCRYPTION KEM).** A signcryption KEM **SC_KEM** = $\{Com, Key_s, Key_r, Encap, Decap, Ver\}$ consists of six algorithms.

- The common parameter generation algorithm *Com* generates parameters shared by all users of **SC_KEM**, such as choice of groups and hash functions. It takes the security parameter $k$ as input, and returns some public information $I$ available to all participants. Notation: $I \xleftarrow{R} Com(k)$.
- The sender key generation algorithm $Key_s$ generates keys for individual users, to be used when signcrypting messages. It takes the public information $I$ as input, and returns a private/public keypair $(x_s, y_s)$. Notation: $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
- The receiver key generation algorithm $Key_s$ generates keys for individual users, to be used when unsigncrypting ciphertexts. It takes the public information $I$ as input, and returns a private/public keypair $(x_r, y_r)$. Notation: $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
- The key encapsulation algorithm *Encap* is used by the sender to generate an authenticated symmetric key. It takes the sender's private key $x_s$, the receiver's public key $y_r$ and a message $m$[1] as input, and outputs a symmetric key $K$ and an encapsulation of that key $e$. Notation: $(K, e) \xleftarrow{R} Encap(x_s, y_r, m)$.
- The key decapsulation algorithm *Decap* is used to recover the symmetric key from an encapsulation. It takes the sender's public key $y_s$, the receiver's private key $x_s$ and an encapsulation $e$ as input, and outputs a symmetric key $K$. Notation: $K \leftarrow Decap(y_s, x_r, e)$.
- The verification algorithm *Ver* is used to verify that an encapsulation corresponds to a given sender, receiver and message. It takes the sender's public key $y_s$, the receiver's private key $x_r$, a message $m$ and an encapsulation $e$ as input, and returns either $\top$ or $\bot$. Notation: $\{\top, \bot\} \leftarrow Ver(y_s, x_r, m, e)$.

The **SC_KEM** must be sound, in the sense that $Decap(y_s, x_r, e)$ should output $K$ whenever $(K, e)$ was the output of $Encap(x_s, y_r, m)$.

**DEFINITION 4.2.2 (SIGNCRYPTION DEM).** A signcryption DEM **SC_DEM** = $\{Encr, Decr\}$ consists of two algorithms. Both algorithms take as auxiliary input a symmetric key $K$. Signcryption DEMs are similar to ordinary DEMs, but are defined separately to indicate that there may be different requirements for a data encapsulation mechanism intended for use in a hybrid signcryption scheme than one used for hybrid encryption.

- The symmetric encryption algorithm *Encr* takes a key $K$ and a message $m$ as input, and outputs a ciphertext $c$. Notation: $c \leftarrow Encr_K(m)$.

[1] This is a major difference from hybrid encryption schemes. Dent shows that the **SC_KEM** needs to act as a signature on $m$ to be insider secure, and hence the encapsulation algorithm needs to take $m$ as input.

- The decryption algorithm *Decr* takes a key $K$ and a ciphertext $c$ as input, and outputs a message $m$. Notation: $m \leftarrow Decr_K(c)$.

The **SC_DEM** must be sound, in the sense that $Decr_K\Big(Encr_K(m)\Big) = m$ for any valid key $K$.

A hybrid signcryption scheme is a signcryption scheme constructed from a signcryption KEM and a signcryption DEM in the following manner:

**Definition 4.2.3 (Hybrid signcryption scheme).** Suppose that **SC_KEM** = {*Com*, $Key_s$, $Key_r$, *Encap*, *Decap*, *Ver*} is a signcryption KEM and that **SC_DEM** = {*Encr*, *Decr*} is a signcryption DEM. If the keys output by *Encap* for a given security parameter $k$ are on the proper form for use with **SC_DEM**, then one may construct a hybrid signcryption scheme by using the *Com*, $Key_s$, and $Key_r$ algorithms from **SC_KEM**, and defining *SC* and *USC* as follows:

- The signcryption algorithm *SC* takes the sender's private key $x_s$, the receiver's public key $y_r$, and a message $m$ as input. It first runs $Encap(x_s, y_s, m)$ to obtain a symmetric key $K$ and an encapsulation $e$. Then it runs $Encr_K(m)$ to obtain a ciphertext $c$, and outputs the pair $(c, e)$ as the signcryptext $\sigma$.
- The unsigncryption algorithm *USC* takes the sender's public key $y_s$, the receiver's private key $x_s$ and a signcryptext $\sigma$ as input. It first parses $\sigma$ into a ciphertext $c$ and an encapsulation $e$. Then it runs $Decap(y_s, x_r, e)$ to obtain a symmetric key $K$, and $Decr_K(c)$ to obtain a message $m$. Finally it runs $Ver(y_s, x_r, m, e)$. If *Ver* returns the symbol $\top$, *USC* outputs $m$; otherwise it outputs $\perp$.

The hybrid signcryption scheme defined above will be sound, provided that the underlying signcryption KEM and DEM are sound.

Regarding the security of a signcryption KEM, Dent states three conditions that he calls IND-CCA2, INP-CCA2 and INT-CCA2. Informally speaking, IND-CCA2 is the requirement that the symmetric key output by the *Encap* algorithm are *ind*istinguishable from a key chosen uniformly at random from $\mathcal{K}$. INP-CCA2 captures the concept that the encapsulation output by *Encap* should be indistinguishable with respect to the *inp*ut message $m$. The notion of INT-CCA2 corresponds to the *int*egrity of encapsulations, in the sense that an adversary can not tamper with or create new encapsulations[2]. The three security notions can be formalised through the following games:

---

Public information:

Signcryption KEM **SC_KEM** = {*Com*, $Key_s$, $Key_r$, *Encap*, *Decap*, *Ver*}.

Primary security parameter $k$, derived security parameter $k_e$.

IND–CCA2–experiment:

$I \xleftarrow{R} Com(k)$.

$(x_s, y_s) \xleftarrow{R} Key_s(I)$.

$(x_r, y_r) \xleftarrow{R} Key_r(I)$.

[2]It is also somewhat of a misnomer, since there is not an adaptive chosen ciphertext attack occurring in the usual sense of the term.

$(m, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$
$(K_0, e^*) \xleftarrow{R} Encap(x_s, y_r, m)$.
$K_1 \xleftarrow{R} \{0,1\}^{k_e}$.
$b \xleftarrow{R} \{0,1\}$.
$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, K_b, e^*; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$.
$\mathcal{A}$ wins if $b' = b$ and $\mathcal{A}_2$ has not asked $e^*$ to *Decap_Oracle*.

INP–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$
$b \xleftarrow{R} \{0,1\}$.
$(K^*, e^*) \xleftarrow{R} Encap(x_s, y_r, m_b)$.
$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, e^*; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$.
$\mathcal{A}$ wins if $b' = b$ and $\mathcal{A}_2$ has not asked $(m_0, e^*)$ or $(m_1, e^*)$ to *Ver_Oracle*.

INT–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$(m^*, e^*) \xleftarrow{R} \mathcal{A}(I, y_s, x_r; Encap\_Oracle)$
$\mathcal{A}$ wins if $\top \leftarrow Ver(x_s, y_r, m^*, e^*)$ and $e^*$ was never returned by *Encap_Oracle* as part of its response to a query $m^*$.

*Encap_Oracle*$(m)$:
Return $Encap(x_s, y_r, m)$.

*Decap_Oracle*$(e)$:
Return $Decap(y_s, x_r, e)$.

*Ver_Oracle*$(m, e)$:
Return $Ver(y_s, x_r, m, e)$.

**Listing 4.3:** Security notions for signcryption KEMs.

Regarding the signcryption DEM, Dent's requirement is that symmetric encryption algorithm is IND-PA secure. It is also required that the decryption algorithm *Decr* is one-to-one[3]. Dent comments that this is a technicality relating to strong unforgeability. To see this, consider an attacker Mallory, who has access to a signcryption oracle and obtained the message/signcryptext pair $m, \sigma$, with $\sigma = (c, e)$. If *Decr* is not one-to-one, it can not be ruled out directly that Mallory is able to find a new ciphertext $c'$ that decrypts to $m$ under the corresponding

[3]This is the case for all symmetric encryption schemes in common use.

secret key $K$. In this case, $(c', e)$ constitutes an existential forgery [Den04, Section 5.4]. Despite the direct connection with strong unforgeability, it is interesting to notice that Dent and also Baek, Steinfeld and Zheng, require this condition in their proofs for *indistinguishability* of signcryption [BSZ02] [Den04].

Dent's main result is that a hybrid signcryption scheme is IND-CCA2 and sUF-CMA secure, whenever the underlying signcryption KEM and DEM satisfy their respective security notions as defined above. The following theorem relates the unforgeability of signcryption to the integrity of the signcryption KEM. It provides a slightly stronger result than that of Dent [Den04], as it gives a tight reduction in both directions.

**Theorem 4.2.4 (Integrity of hybrid construction).** *Let* **SC** = {*Gen, Key*$_s$*, Key*$_r$*, SC, USC*} *be the hybrid signcryption scheme constructed from a signcryption KEM* **SC_KEM** = {*Gen, Key*$_s$*, Key*$_r$*, Encap, Decap*} *and a signcryption DEM* **SC_DEM** = {*Encr, Decr*} *whose decryption function is one-to-one. Then* **SC** *is sUF-CMA secure if and only if* **SC_KEM** *is INT-CCA2 secure. Concretely, there exist sUF-CMA adversaries* $\mathcal{A}$ *and* $\mathcal{A}'$ *against* **SC** *and INT-CCA2 adversaries* $\mathcal{B}$ *and* $\mathcal{B}'$ *against* **SC_KEM**, *such that*

$$\begin{aligned} Adv_{sUF-CMA,\mathcal{A}}(k, t, q_{sc}) &\leq Adv_{INT-CCA2,\mathcal{B}}(k, O(t) + t_o, q_{sc}), \\ Adv_{INT-CCA2,\mathcal{B}'}(k, t', q_{enc}) &\leq Adv_{sUF-CMA,\mathcal{A}'}(k, O(t') + t'_o, q_{enc}). \end{aligned} \tag{4.1}$$

*In the above equation,* $q_{sc}$ *is the number of signcryption oracle queries asked by* $\mathcal{A}$*, and* $q_{enc}$ *the number of encapsulation oracle queries asked by* $\mathcal{B}'$*. The running times* $t_o$ *and* $t'_o$ *denote the time required by* $\mathcal{B}$ *and* $\mathcal{A}'$ *to simulate* $q_{sc}$ *and* $q_{enc}$ *oracle queries, respectively.*

*Proof.* The theorem is proved through the explicit construction of the adversary algorithms $\mathcal{B}$ and $\mathcal{A}'$. These algorithms use black-box algorithms $\mathcal{A}$ and $\mathcal{B}'$ to do the difficult work.

Let $\mathcal{A}$ be a sUF-CMA-adversary against **SC** that asks at most $q_{sc}$ queries to its signcryption oracle. One may then construct an INT-CCA2 adversary $\mathcal{B}$ against **SC_KEM** that asks at most $q_{sc}$ queries to its encapsulation oracle in the following manner:

---

$\mathcal{B}(I, y_s, x_r; Encap\_Oracle)$ :
  $\sigma \xleftarrow{R} \mathcal{A}(I, y_s, x_r; SC\_Oracle)$ .
  Parse $\sigma \leftarrow (c, e)$ .
  $K \leftarrow Decap(y_s, x_r, e)$ .
  $m \leftarrow Decr_K(c)$ .
  Return $(m, e)$ .

$SC\_Oracle(m)$ :
  $(K, e) \leftarrow Encap\_Oracle(m)$ .
  $c \leftarrow Encr_K(m)$ .
  Return $(c, e)$ .

---

**Listing 4.4:** Construction of INT-CCA2 adversary against **SC_KEM**.

The adversary $\mathcal{A}$ will not be able to distinguish whether it is run in a real attack game or by $\mathcal{B}$, since its runtime environment is simulated perfectly. Furthermore, whenever $\mathcal{A}$ is

successful it returns a valid forgery $\sigma$ that unsigncrypts to a message $m$, such that $\sigma$ has not been a reply from *SC_Oracle* to the query $m$. Because **SC_DEM** is one-to-one, this implies that $e$ has not been a reply from *Encap_Oracle* to the query $m$ either. Hence $(m, e)$ is a valid forgery. Finally, the runtime of $\mathcal{B}$ is given by the total time used by $\mathcal{A}$, plus the time required to simulate $q_{sc}$ oracle queries and perform the final forgery. It follows that $\mathcal{B}$ is an efficient adversary with respect to runtime and success rate whenever $\mathcal{A}$ is.

To prove the reverse implication, let $\mathcal{B}'$ be an INT-CCA2 adversary against **SC_KEM** that asks $q_{enc}$ oracle queries to its encapsulation oracle. The following sUF-CCA2 adversary $\mathcal{A}'$ breaks the security of **SC**, while asking $q_{enc}$ queries to its signcryption oracle:

---

$\mathcal{A}'(I, y_s, x_r; SC\_Oracle)$ :
  $(m, e) \xleftarrow{R} \mathcal{B}'(I, y_s, x_r; Encap\_Oracle)$ .
  $K \leftarrow Decap(y_s, x_r, e)$ .
  $c \leftarrow Encr_K(m)$ .
  Return $\sigma = (c, e)$ .

$Encap\_Oracle(m)$ :
  $\sigma \xleftarrow{R} SC\_Oracle(m)$ .
  Parse $\sigma \leftarrow (c, e)$ .
  Return $e$ .

---

**Listing 4.5:** Construction of sUF-CMA adversary against $\mathcal{SC}$.

The runtime environment of $\mathcal{B}'$ is again simulated perfectly. Furthermore, whenever $\mathcal{B}'$ returns a valid forgery $(m, e)$ such that $e$ has not been reply from *Encap_Oracle* on the message $m$, it is clear from the construction of *Encap_Oracle* that $e$ was never part of the reply from *SC_Oracle* either. Hence, $\mathcal{A}'$ wins whenever $\mathcal{B}'$ does, with runtime equal to the runtime of $\mathcal{B}'$ plus the time required to simulate $q'$ oracle queries and perform the final forgery. ■

The previous theorem demonstrates that the notion of sUF-CMA for signcryption is captured extremely well by the notion of INT-CMA for signcryption KEMs. In a very real sense, an INT-CCA2 secure signcryption KEM is acting as a signature scheme on the input message, as well as providing the expected key encapsulation service. This implies that the corresponding signcryption DEM is only required to keep messages confidential, and need not provide any kind of message authentication services. The next theorem gives a similar security result for the confidentiality of hybrid signcryption.

**THEOREM 4.2.5.** *Let* **SC** $= \{Gen, Key_s, Key_r, SC, USC\}$ *be a hybrid signcryption scheme constructed from* **SC_KEM** $= \{Gen, Key_s, Key_r, Encap, Decap\}$ *and* **SC_DEM** $= \{Encr, Decr\}$. *Suppose that there exists an adversary* $\mathcal{A}$ *that wins the IND-CCA2 game for* **SC** *with running time* $t$, *that asks* $q_{sc}$ *queries to its signcryption oracle and* $q_{usc}$ *queries to its unsigncryption oracle. Then there exists adversaries* $\mathcal{B}, \mathcal{C}, \mathcal{D}$ *against* **SC_KEM** *and* $\mathcal{E}$ *against* **SC_DEM**, *such that*

$$\begin{aligned} Adv_{SC-IND-CCA2,\mathcal{A}}(k, t, q_{sc}, q_{usc}) \leq \\ 2\, Adv_{IND-CCA2,\mathcal{B}}(k, t_1, q_{sc}, q_{usc}, q_{usc}) + Adv_{INT-CCA2,\mathcal{C}}(k, t_2, q_{sc}) \\ +2\, Adv_{INP-CCA2,\mathcal{D}}(k, t_3, q_{sc}, q_{usc}, q_{usc}) + Adv_{IND-PA,\mathcal{E}}(k, t_4). \end{aligned} \tag{4.2}$$

*The adversaries $\mathcal{B}$ and $\mathcal{D}$ ask $q_{sc}$ queries to their encapsulation oracle, and at most $q_{usc}$ queries to their decapsulation and verification oracles. Adversary $C$ asks $q_{sc}$ encapsulation oracle queries. The running times of each algorithm are $O(t)$, plus the time needed by that algorithm to simulate $q_{sc}$ signcryption oracle queries and $q_{usc}$ unsigncryption oracle queries*[4].

*Proof.* The proof uses standard game hopping techniques, with the three first adversaries corresponding to game transitions, and the final adversary to the advantage of $\mathcal{A}$ in the final game. It is omitted due to its length, but may be found in [Den04, Section 5]. ■

Theorems 4.2.4 and 4.2.5 provide a solid theoretical foundation for the analysis of hybrid signcryption schemes in Dent's framework. They demonstrate that the generic composition of secure signcryption KEM and secure signcryption DEM is itself secure under Dent's security models.

## 4.3 Multi-user security

Whereas the basic security models for signature and encryption schemes only consider one user, the basic security models discussed in Chapter 4.1 as well as the hybrid model in Chapter 4.2 consider communication between two fixed users. This is not necessarily a good model for the actual real-world usage of signcryption, where a large number of active users may coexist. Additional capabilities might be available to an adversary who can control several users simultaneously, or create new legitimate users. The use of a flexible unsigncryption oracle by Baek, Steinfeld and Zheng is one attempt at capturing some of this behaviour with respect to the confidentiality of signcryption, as is the proposed extension of their model where the adversary may also use the signcryption oracle flexibly [BSZ02]. The model used by Malone-Lee in [ML04b] goes even further, and allows oracle access to signcryption and unsigncryption between *any* pair of legitimate users. Dent argues that this may still not be general enough, since an adversary might plausibly gain further advantage by learning the private keys of trusted users, for example through bribery or theft [Den04]. There does not appear to be any consensus as to which security model is best suited to capturing the possible actions of an adversary, while being sufficiently restricted that practical security proofs may be performed.

With regards to authenticity and integrity of signcryptions, it would appear reasonable to permit the adversary to attempt to perform existential forgery of a message from Alice to *anyone*, rather than to a fixed receiver. A possible notion of "multiparty unforgeability" (MUF) may thus be proposed. In a MUF-experiment, the adversary takes Alice's public sending key $y_s$ as input, and wins whenever he outputs a valid receiver's keypair $(x_r, y_r)$ and a valid signcryption $\sigma$, such that $\perp \neq m \leftarrow USC(y_s, x_r, \sigma)$. The adversary would need access to a flexible signcryption oracle that takes a public receiver key $y_r'$ and a message $m$ as input, and outputs Alice's signcryption of the message from herself to the specified receiver. MUF-CMA security of signcryption might thus be examined under either regular or strong notions of unforgeability. The author does not know of any analysis having been attempted under this model. It is thus not clear whether practical security results may be achieved in such a permissive model.

[4] This may be done in different ways by the different adversaries, but is in all cases $O(q_{sc} + q_{usc})$.

# 5

# Security of Zheng's signcryption scheme

This chapter contains reworkings of the security proofs for authenticity and confidentiality of Zheng's signcryption scheme performed by Baek, Steinfeld and Zheng [BSZ02]. The intention of the chapter is to make their results more accessible to readers by presenting them in a structured manner.

## 5.1 Authenticity of Zheng's signcryption scheme

Although the original paper by Baek, Steinfeld and Zheng gives a security bound for the UF-CMA security of Zheng's signcryption scheme, their paper omit the details of the proof [BSZ02].

**Claim 5.1.1 (UF-CMA security of $\mathcal{SC}$).** If the discrete logarithm problem is hard, then $\mathcal{SC}$ is secure against existential forgery in the Random Oracle Model with the following bound:

$$Adv_{UF-CMA}(k, t, q_{sc}, q_g, q_h) \leq 2q_h \cdot \left(Adv_{DL}(k, t^*)\right)^{\frac{1}{2}} + \frac{1}{q}. \tag{5.1}$$

In the above expression, the discrete logarithm problem is solved with respect to the sender's public key $y_s$. The time required to create the forgery is denoted by $t$, while $q_{sc}, q_g$ and $q_h$ denote the number of queries to the signcryption oracle and random oracles, respectively. The time used to compute the discrete logarithm is denoted $t^* = O(t + t_{sc} + t_v + t_c)$, where $t_{sc}$ is the time required to simulate $q_{sc}$ queries to the signcryption oracle, $t_v$ is the time used by the verification step in the identification protocol associated to SDSS1, and $t_c$ is the time required for the final computation of $x_s$.

Although few details are given in the original paper, the proof itself can be deduced to have been performed in three steps. The first step is to prove that existential forgery of a ciphertext in $\mathcal{SC}$ leads to existential forgery of SDSS1 signatures. Proving that a signature scheme is existentially unforgeable is a problem with much more applicable theory than the unforgeability of signcryption schemes. As indicated in [BSZ02], the second step of the security proof is to apply the "ID reduction lemma" [OO98] to SDSS1. It is used to relate the success rate of an adversary that forges SDSS1 signatures, with that of a cheater in an associated identification protocol. Finally, the heavy row lemma [OO98] [ML04b] is used to turn an adversary breaking the identification protocol into an algorithm for computing discrete logarithms.

**Theorem 5.1.2 (Security of $\mathcal{SC}$ relative to SDSS1).** *Assume that there exists an adversary $\mathcal{A}$ that wins the UF-CMA game against $\mathcal{SC}$ in time $t$, using $q_{sc}$ queries to its signing oracle and $q_g$ and*

*$q_h$ queries to the respective random oracles. Then there exists an algorithm $\mathcal{B}$ that wins the UF-CMA game against the SDSS1 signature scheme such that*

$$Adv_{UF-CMA,\mathcal{A}}(k,t,q_{sc},q_g,q_h) \leq Adv_{UF-CMA,\mathcal{B}}(k,t',q_{sc},q_{sc}+q_h) + \frac{2q_h + q_{sc}\cdot(q_{sc}-1)}{2q}. \quad (5.2)$$

*The algorithm $\mathcal{B}$ asks asks $q_{sc}$ queries to its signing oracle and $q_{sc} + q_h$ queries to its random oracle. It runs in time $t' = O(t) + t_o$, where $t_o$ is the time it requires to simulate $q_{sc}$ signcryption queries and $q_h$ random oracle queries.*

*Proof.* The following listing specifies the initial UF-CMA game against $\mathcal{SC}$ in its entirety. Note the usage of random oracles in place of the cryptographic hash functions $\mathcal{G}$ and $\mathcal{H}$. The oracles are simulated by lazy evaluation using lists $L_G$ and $L_H$ to maintain state between queries, as discussed in Remark 2.6.3.

---

Public information: Security parameters $k$, $k_q$, $k_e$.

UF–CMA Experiment against SC:
- $I \xleftarrow{R} Com(k)$.
- $(x_s, y_s) \xleftarrow{R} Key_s$.
- $(x_r, y_r) \xleftarrow{R} Key_r$.
- $bind \leftarrow y_s || y_r$.
- $(m^*, \sigma^*) \xleftarrow{R} \mathcal{A}(I, y_s, x_r; SC\_Oracle, G\_Oracle, H\_Oracle)$.
- $\mathcal{A}$ wins if $m^* \leftarrow USC(y_s, x_r, \sigma^*)$ and $m^*$ was never a query to *SC_Oracle*.

*SC_Oracle*$(m)$:
- Return $SC(x_s, y_r, m)$.

*G_Oracle*$(\kappa)$:
- If $(\kappa, K)$ is in the list $L_G$:
  - Return $K$.
- Else:
  - $K \xleftarrow{R} \{0,1\}^{k_e}$.
  - Add $(\kappa, K)$ to $L_G$.
  - Return $K$.

*H_Oracle*$(m||bind||\kappa)$:
- If $(m||bind||\kappa, r)$ is in the list $L_H$:
  - Return $r$.
- Else:
  - $r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
  - Add $(m||bind||\kappa, r)$ to $L_H$.
  - Return $r$.

---

**Listing 5.1:** Game 0: UF-CMA game against $\mathcal{SC}$.

Next, consider the following algorithm $\mathcal{B}$, that plays the UF-CMA game against SDSS1 and uses $\mathcal{A}$ as a subroutine.

---

Public information: Security parameters $k$, $k_q$, $k_e$.

$\mathcal{B}(I, y_s; Sign\_Oracle, H\_Oracle)$:
  $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
  $bind \leftarrow y_s || y_r$.
  $(m^*, \sigma^*) \xleftarrow{R} \mathcal{A}(I, y_s, x_r; SC\_Sim, G\_Sim, H\_Sim)$.
  If $m^* \leftarrow USC(y_s, x_r, \sigma^*)$ and $m^*$ was never queried to $SC\_Oracle$:
    Parse $(c^*, r^*, s^*) \leftarrow \sigma^*$.
    Return $\left(m^*, (r^*, s^*)\right)$.
  Else return $\perp$.

$SC\_Sim(m)$:
  $(r, s) \leftarrow Sign\_Oracle(m)$.
  $\kappa \leftarrow (y_s \cdot g^r)^{s \cdot x_r} \mod p$.
  $K \leftarrow G\_Sim(\kappa)$.
  $c \leftarrow Encr_K(m)$.
  $\sigma \leftarrow (c, r, s)$.
  Add $(m||bind||\kappa, r)$ to $L_H$.
  Return $\sigma$.

$G\_Sim(\kappa)$:
  If $(\kappa, K)$ is in the list $L_G$:
    Return $K$.
  Else:
    $K \xleftarrow{R} \{0,1\}^{k_e}$.
    Add $(\kappa, K)$ to $L_G$.
    Return $K$.

$H\_Sim(m||bind||\kappa)$:
  $\kappa' \leftarrow \kappa^{(x_r^{-1})} \mod p$.
  $r \leftarrow H\_Oracle(m||bind||\kappa')$.
  Add $(m||bind||\kappa, r)$ to $L_H$.
  Return $r$.

---

**Listing 5.2:** Game 1: UF-CMA game against SDSS1.

From the point of view of $\mathcal{A}$, Game 0 and Game 1 are almost identical, and the transition between them is based on a failure event. The keys given as input to $\mathcal{A}$ are independent and randomly distributed in both games. Likewise, every value returned from $H\_Sim$ in Game 1 will be distributed identically to those from $H\_Oracle$, independently of the input. Indeed, the only event that may cause Game 0 and Game 1 to differ, is that $SC\_Sim$ returns a value $r$ whose preimage $m||bind||\kappa$ has already been assigned a value in $L_H$. To bound the probability of this occuring, consider the "worst" scenario that may occur: $\mathcal{A}$ asks $q_h$ queries of the

form $m||bind||*$ to *H_Sim*, before querying *SC_Sim* with the same $m$, $q_{sc}$ times. The probability of the first query to *SC_Sim* causing an error is equal to $\frac{q_h}{q}$. For each query that does not cause the failure event to occur, another value $m||bind||\kappa$ may have been fixed by *SC_Sim*. Hence, summing over $q_{sc}$ queries, the total probability that no errors have occurred is equal to $\epsilon = \frac{2q_h + q_{sc}\cdot(q_{sc}-1)}{2q}$. This is negligible with respect to the security parameter $k_q$. By Lemma 2.5.2, $\left|Pr[\mathcal{A} \text{ wins Game 0}] - Pr[\mathcal{A} \text{ wins Game 1}]\right| \leq \epsilon$. Due to the way that *SC_Sim* and *H_Sim* are constructed, $\mathcal{B}$ wins the UF-CMA game against SDSS1 whenever $\mathcal{A}$ wins in Game 1. This is because $(r, s)$ will always be a valid SDSS1 signature on $m$, and the only time that $m$ is asked to the *Sign_Oracle* is when it is also asked to *SC_Oracle*. The number of signing queries asked by $\mathcal{B}$ is equal to $q_{sc}$, and the number of random oracle queries $q_{sc} + q_h$. Finally, the running time of $\mathcal{B}$ is precisely $O(t)$ plus the time needed to run the oracle simulations. ■

The remainder of this section follows the proof of unforgeability of the Schnorr signature scheme in [OO98]. In order to proceed with a reduction from an adversary forging SDSS1 to an adversary cheating in the corresponding identification protocol, it is necessary to state said identification protocol.

**Definition 5.1.3 (SDSS1-ID).** The identification scheme associated with SDSS1 through the Fiat-Shamir heuristic (as discussed in Chapter 2.6) uses the common parameter and key generation algorithm *Com* and $Key_s$ from SDSS1, together with the following three-move zero-knowledge protocol.

- Commitment: Peggy chooses a random value $n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$ and computes a commitment value $\kappa \xleftarrow{R} g^n \mod p$. She sends $\kappa$ to Victor.
- Challenge: Victor generates a random challenge $r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$ and sends it to Peggy.
- Response: Peggy uses her private key $x_s$ to compute a response $s \leftarrow n/(x_s + r) \mod q$, and sends it to Victor.
- Verification: Victor uses Peggy's public key $y_s$ to check whether $(y_s + g^r)^s = \kappa$.

The identification protocol fails (similarly to $\mathcal{SC}$ and SDSS1) when $r = -x_s$. The probability of this occuring is negligible with respect to the security parameter $k_q$.

In [OO98] the identification protocol under consideration is assumed to be *perfect* zero knowledge with respect to a honest verifier. This means that there exists a simulator algorithm **S** that can simulate protocol exchanges between Peggy and Victor, such that all protocol transcripts have distributions that are identical to the real protocol run with a honest Victor [Gol01] [MVO96, Defn. 10.20]. This simulability is known as the zero-knowledge property; the verifier gains no knowledge from the exchange that he could not have computed himself. However, in practice a weaker requirement must usually be substituted: that the **S** only produces distributions that are indistinguishable in polynomial time by a probabilistic algorithm. This property is called *computational* zero knowledge. The following algorithm can be used to simulate SDSS1-ID.

---

Public information:
Security parameters $k$, $k_q$.

$k$–bit prime $p$, $k_q$–bit prime $q$ such that $q|(p-1)$.
Element $g \in \mathbb{Z}_p$ of order $q$.

Simulator $\mathbf{S}(y_s)$:
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa \leftarrow (y_s \cdot g^r)^s \mod p$.
$\tau' \leftarrow (\kappa, r, s)$.
Return $\tau'$.

**Listing 5.3:** ID transcript simulator

**Proposition 5.1.4.** *SDSS1-ID has the computational zero knowledge property.*

*Proof.* Consider the probability distribution of a valid protocol transcript $\tau$, compared to a simulated transcript $\tau'$. In the original protocol, $\kappa$ is distributed uniformly at random, because $n$ is distributed uniformly at random and $g$ is of order $q$ in $\mathbb{Z}/q\mathbb{Z}$. The challenge $r$ is distributed uniformly at random over $\mathbb{Z}/q\mathbb{Z} \setminus \{-x_s\}$, since the protocol fails and no transcript is produced when $r$ takes the value of $-x_s$. Finally, the response $s$ is distributed uniformly on $\mathbb{Z}/q\mathbb{Z}$, as the product of an uniformly distributed element and a non-zero element constitutes a permutation. In the simulated protocol, it is $r$ and $s$ that are uniformly and independently distributed over $\mathbb{Z}/q\mathbb{Z}$, while $\kappa$ differs from the uniform distribution and takes the value 1 whenever $r = -x_s$. One may notice that the dependency relation between $\kappa$, $r$ and $s$ is identical in both transcripts, since $\kappa = (y_s \cdot g^r)^s \mod p$ for both $\tau$ and $\tau'$. Whether one considers $\kappa$ or $s$ as the dependent variable is a question of what is most convenient, since the causality involved is not known when viewing a transcript in retrospect.

The statistical distance between two probability distributions $X$ and $Y$ on a finite set $S$ is defined as $Dist(X,Y) = \frac{1}{2}\sum_{s\in S}\left|Pr[X=s] - Pr[Y=s]\right|$. If the distance between the distributions of $\tau$ and $\tau'$ over the set $\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$[1] is negligible with respect to the security parameter $k_q$, then the distributions are said to be *statistically indistinguishable*. This is a strong result, which implies that the distributions in question are also computationally indistinguishable [Gol01].

In the $q \cdot (q-1)$ cases where $r \neq -x_s$, the probability of a specific transcript $(\kappa, r, s)$ being output by the real protocol is $\frac{1}{q\cdot(q-1)}$, while the the probability of those same values being output by $\mathbf{S}$ is $\frac{1}{q^2}$. Their difference is hence $\frac{1}{q^2\cdot(q-1)}$. In the $q$ possible cases where $r = -x_s$, $Pr\left[\tau = (*, -x_s, *)\right] = 0$. However, $Pr\left[\tau' = (*, -x_s, *)\right]$ is again $\frac{1}{q^2}$. Hence, the statistical distance between the distributions is given by

$$Dist(\tau, \tau') = \frac{1}{2}\left(q \cdot (q-1) \cdot \frac{1}{q^2 \cdot (q-1)} + q \cdot \frac{1}{q^2}\right) = \frac{1}{q}. \tag{5.3}$$

This is negligible in $k_q$. Since the distributions of $\tau$ and $\tau'$ are statistically indistinguishable and the dependence relation between the variables are identical in both cases, one may conclude that SDSS1-ID has computational zero-knowledge property. ∎

[1] Viewing $\kappa$ as a dependent variable of $r$, $s$ and a fixed keypair.

The assumption that SDSS1-ID is perfect zero knowledge does not appear to be valid, assuming that there is no better way of constructing **S** than above. This is not discussed in [BSZ02], which indeed suggests using the ID reduction technique for Zheng's scheme. Neither is the assumption discussed in the original article [OO98], even though a proof of security for Schnorr's signature scheme (which is very similar to SDSS1, and whose associated identification scheme also appears to be computational zero knowledge) is used as an example. The strength of this assumption must therefore be taken into account when considering the overall strength of the security proof.

**Lemma 5.1.5 (ID Reduction Lemma).** *If there exists an UF-CMA adversary $\mathcal{B}$ against SDSS1 with advantage $Adv_{UF-CMA,\mathcal{B}}(k, t', q_s, q_h) = \epsilon'$, then there exists an algorithm $C$ that fools Victor in the corresponding identification protocol with advantage $Adv_{ID,C}(k, t'') = \epsilon''$. In the above reduction, $q_s$ and $q_h$ are the number of oracle queries used by $\mathcal{A}$, $t'' = O(t') + t_s$, where $t_s$ is the time needed to simulate $q_s$ signatures, and $\epsilon'' = \frac{1}{q_h}\left(\epsilon' - \frac{q_h \cdot q_s + 1}{q}\right)$.*

*Proof.* The proof is given in [OO98, Section 3]. ∎

The final step of the proof is to use an algorithm that is able to cheat Victor in the SDSS1-ID protocol, to compute the discrete logarithm of Peggy's public key. To do this, a simple result called the Heavy Row Lemma [OO98] [ML04b] is used. It should also be pointed out that this lemma is in fact a special case of the better-known Splitting Lemma used by Pointcheval and Stern [PS96] [PS00]. The setting in the following definitions is one where an adversary $C$ is run repeatedly under controlled circumstances in an experiment against the security of the identification protocol SDSS1-ID.

**Definition 5.1.6 (Boolean matrix of random tapes).** Consider a hypothetical matrix $M$ whose rows consists of all possible random choices made by $C$, and whose columns consists of all possible random choices made by Victor. Each row then corresponds to a unique random tape that may be used by $C$, and similarly for each column. Let each matrix entry be either $\perp$ if Victor rejects $C$'s proof, or $\top$ if $C$ wins the game and fools Victor[2].

**Definition 5.1.7 (Heavy row).** A row in $M$ is called *heavy* if the fraction of $\top$'s along a row is greater than or equal to $\epsilon/2$, where $\epsilon$ is the success rate of $\mathcal{A}$.

**Lemma 5.1.8 (Heavy row lemma).** *Let $M$ be a boolean matrix as defined above. If a given entry of $M$ is equal to $\top$, then the probability that it lies in a heavy row is at least $\frac{1}{2}$.*

*Proof.* The result follows from simple manipulation of probabilities:

$$\begin{aligned} Pr[\text{Heavy row} \mid \text{Entry is } \top] &= 1 - Pr[\text{Nonheavy row} \mid \text{Entry is } \top] \\ &= 1 - \frac{Pr[\text{Nonheavy row } \wedge \text{ Entry is } \top]}{Pr[\text{Entry is } \top]} \\ &> 1 - \frac{\frac{1}{2} \cdot Pr[C \text{ wins}]}{Pr[C\text{wins}]} = \frac{1}{2}. \end{aligned} \tag{5.4}$$

The final transition follows from the definition of heavy row. If the row is *not* heavy, then the fraction of $\top$'s along that row must be less than $\frac{\epsilon}{2}$. From the definition of heavy row, the probability that this occurs is strictly less than half the probability that $C$ wins. ∎

[2]Since algorithms are assumed to terminate, the matrix $M$ may safely be assumed finite. However, the following results concerning $M$ are easily be extended to sections of a general product space $X \times Y$.

The basic idea to be applied is that $C$ is simply a black-box probabilistic algorithm. It may be thus run under controlled circumstances by someone controlling its runtime environment, including the random tape. If it is possible to run $C$ in such a manner that it would have fooled Victor twice with the same commitment $\kappa$ but with a different challenge $r$, the information revealed can be used to extract Peggy's private key $x_s$.

**Proposition 5.1.9.** *Suppose that one is able to procure two valid transcripts of the identification protocol SDSS1-ID, in which Peggy has given the same commitment value $\kappa = g^n \mod p$, and then replied with two valid responses $s_0, s_1$ to two different challenges $r_0 \neq r_1$. Then one may easily compute the discrete logarithm of Peggy's public key.*

*Proof.* From the definition of SDSS1-ID, a valid protocol transcript means that $s_0 = n/(x_s + r_0) \mod q$ and $s_1 = n/(x_s + r_1) \mod q$. Because $r_0 \neq r_1$, it follows that $s_0 \neq s_1$. Furthermore, one may assume that $x_s + r_0 \neq 0 \neq x_s + r_1$, since $\mathcal{A}$ does not fool Victor when the protocol fails[3]. The only unknowns are $n$ and $x_s$, and slight manipulation yields

$$\begin{aligned} n = s_0 \cdot (x_s + r_0) = s_1 \cdot (x_s + r_1) \mod q, \\ x_s \cdot (s_0 - s_1) = s_1 \cdot r_1 - s_0 \cdot r_0 \mod q. \end{aligned} \tag{5.5}$$

Because $s_0 \neq s_1$, it follows that $s_1 \cdot r_1 - s_0 \cdot r_0 \neq 0 \mod q$. Hence Peggy's private key $x_s$ may be computed as

$$x_s = \frac{s_1 \cdot r_1 - s_0 \cdot r_0}{s_0 - s_1} \mod q. \tag{5.6}$$

■

Combining Lemma 5.1.8 and Proposition 5.1.9 allows one to use $C$ to compute the discrete logarithm of Peggy's public key $y_s$.

**Theorem 5.1.10 (Security of ID scheme).** *Suppose that there exists an algorithm $C$ such that $Adv_{\mathcal{ID},C}(k, t'') = \epsilon'' \geq \frac{4}{q}$. Then there exists an algorithm $\mathcal{D}$ that computes the discrete logarithm of a public key $y = g^x \mod p$ with $Adv_{DL,\mathcal{D}}(k, 2t'' + t_c) \geq \frac{(\epsilon'')^2}{4}$, where $t_c$ is the time required for the final computation of the discrete logarithm $x$ from $s_0, r_0, s_1$ and $r_1$.*

*Proof.* Consider an algorithm $\mathcal{D}$ that runs $C$ with the specified public key $y$, and a random challenge. This results in a valid protocol forgery with probability $\epsilon''$. The probability that the random tape provided to $C$ is associated with a heavy row of the boolean matrix $M$ is at least $\frac{1}{2}$, by the heavy row lemma. Therefore, $C$ may be run again with the same input and random tape[4], but with a different random response to its initial commitment. If the row in question indeed is heavy, the probability of the second experiment succeeding is $\frac{\epsilon''}{2}$. Hence, the probability of obtaining two collisions is $\frac{(\epsilon'')^2}{4}$. By Proposition 5.1.8, the discrete logarithm of $x$ can be computed in constant time $t_c$ whenever this occurs. ■

**Remark 5.1.11.** The algorithm $\mathcal{D}$ runs $C$ twice, and has a success probability that is quadratic in that of $C$. Alternately, one may perform what essentially constitutes a randomised search of $M$ to find a heavy row. Running $C$ a total of $O(\frac{1}{\epsilon''})$ times, one will succeed at finding the discrete logarithm of $y$ with fixed probability greater than $\frac{1}{2}\left(1 - \frac{1}{e}\right)^2$, where $e$ is the base of the natural logarithm [OO98].

[3] Although it would lead to $\mathcal{A}$ knowing $-x_s$, and hence being able to defeat the protocol in future exchanges if it is allowed to maintain state information between successive trials.

[4] This is sometimes called "rewinding the random tape".

Putting together the results of this section, a bound for the UF-CMA security of $\mathcal{SC}$ may be stated.

**Theorem 5.1.12 (UF-CMA security of $\mathcal{SC}$).** *If the discrete logarithm problem is hard, then $\mathcal{SC}$ is secure against existential forgery in the random oracle model with the following bound:*

$$Adv_{UF-CMA}(k, t, q_{sc}, q_g, q_h) \leq 2q_h \cdot \left(Adv_{DL}(k, t^*)\right)^{\frac{1}{2}} + \frac{2q_h \cdot (q_{sc} + 1) + q_{sc} \cdot (3q_{sc} - 1) + 2}{2q}. \quad (5.7)$$

*In the above expression, $t$ is the time required to forge a valid signcryption, and $q_{sc}, q_g$ and $q_h$ are the number of oracle queries used in the process. The time $t^* = O(t + t_o + t_s + t_c)$, where $t_o$ is the time needed to simulate $q_{sc} + q_h$ oracle queries, $t_s$ is the time needed to simulate $q_{sc}$ SDSS1 signatures, and $t_c$ is the time needed for the final computation of $x_s$.*

*Proof.* The result is obtained by combining Theorem 5.1.2, Lemma 5.1.5 and Theorem 5.1.10. It is important to remember that the number of random oracle queries asked by the adversary against SDSS1 in Lemma 5.1.5 is in fact equal to $q_h + q_{sc}$, due to the reduction in Theorem 5.1.2. ∎

The security bound obtained in Theorem 5.1.12 is slightly weaker than the original bound as stated in Claim 5.1.1. With regard to running times, the terms $t_{sc}$ and $t_v$ in Claim 5.1.1 are defined to be the simulation time for $q_{sc}$ signcryptexts, and the "time for verification in the identification scheme". In contrast, $t_o$ is the total time needed to simulate $q_{sc}$+$q_h$ random oracle queries in addition to $q_{sc}$ signcryption queries. Furthermore, the term $t_s$ comes from the ID reduction lemma, and corresponds to the simulation of $q_{sc}$ SDSS1 signatures. It would appear that the original paper alludes to a more efficient reduction from $\mathcal{SC}$ to SDSS1. Regarding the probability bound itself, the term corresponding to the number of oracle queries performed by the original adversary against $\mathcal{SC}$ has been expanded. A plausible conclusion is that the reduction used by Baek, Steinfeld and Zheng uses only a single signcryption oracle query in the reduction from $\mathcal{SC}$ to SDSS1.

## 5.2 Confidentiality of Zheng's signcryption scheme

Baek, Steinfeld and Zheng also proved that Zheng's signcryption scheme is secure in the FUO-IND-CCA2 model relative to the Gap Diffie-Hellman problem. Their stated result is as follows [BSZ02]:

**Claim 5.2.1.** If the Gap Diffie-Hellman problem is hard and the symmetric encryption scheme $\mathbf{E} = \{Encr, Decr\}$ used is IND-CPA secure, then Zheng's signcryption scheme $\mathcal{SC}$ is secure in the random oracle model with the following bound[5]:

$$Adv_{FUO-IND-CCA2}(k, t, q_{sc}, q_{usc}, q_g, q_h) \leq$$
$$2 \cdot Adv_{GDH}(k, t_1, q_{ddh}) + Adv_{IND-CPA}(k_e, t_2, 0) + \frac{q_{sc}(q_g + q_h + 1) + q_{usc}}{2q}. \quad (5.8)$$

In the above equation, $q_{sc}$, $q_{usc}$, $q_g$ and $q_h$ are the number of oracle queries to the signcryption, flexible unsigncryption and random oracles, while $q_{ddh} = O(q_{sc} + q_g + q_h)$ is the number of

[5]Due to an unusually definition of the advantage function for IND-games used by the original authors, the bound is stated differently in [BSZ02].

queries to the Decisional Diffie-Hellman oracle. The parameters $k_e$ and $k_q$ are the security parameters for **E** and $q$ respectively. The execution times $t_1 = O(t + t_g + t_h + t_{sc} + t_{usc})$ and $t_2 = O(t_1)$ are dependent on the simulation times for the four oracles.

**Remark 5.2.2.** The IND-CPA advantage against **E** is taken over adversaries that ask 0 queries to its chosen plaintext oracle. Since the oracle is not used, it is acting as a completely passive attacker, and the weaker IND-PA security notion may be used instead.

The following listing specifies the original FUO-IND-CCA2 game against $\mathcal{SC}$ in full. Again, random oracles are used in place of the cryptographic hash functions $\mathcal{G}$ and $\mathcal{H}$.

---

Public information:
    Security parameters $k$, $k_q$, $k_e$.

FUO–IND–CCA2–experiment:
    $I \xleftarrow{R} Com(k)$.
    $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
    $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
    $bind \leftarrow y_s || y_r$.
    $(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle, G\_Oracle, H\_Oracle)$.

    $b \xleftarrow{R} \{0,1\}$.
    $n^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
    $\kappa^* \leftarrow g^{n^*} \mod p$.
    $r^* \leftarrow H\_Oracle(m||bind||\kappa^*)$.
    $s^* \leftarrow n^*/(x_s + r^*) \mod q$.
    $K^* \leftarrow G\_Oracle(\kappa^*)$.
    $c^* \leftarrow Encr_{K^*}(m_b)$.
    $\sigma^* \leftarrow (c^*, r^*, s^*)$.

    $b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle, G\_Oracle, H\_Oracle)$.
    $\mathcal{A}$ wins if $b = b'$ and $(y_s, \sigma^*)$ has not been asked to $USC\_Oracle$.

$SC\_Oracle(m)$:
    Return $SC(x_s, y_r, m)$.

$USC\_Oracle(y'_s, \sigma)$:
    Return $USC(y'_s, x_r, \sigma)$.

$G\_Oracle(\kappa)$:
    If $(\kappa, K)$ is in the list $L_G$:
        Return $K$.
    Else:
        $K \xleftarrow{R} \{0,1\}^{k_e}$.
        Add $(\kappa, K)$ to $L_G$.

```
        Return K.

H_Oracle(m||bind||κ):
    If (m||bind||κ, r) is in the list L_H:
        Return r.
    Else:
        r ←R Z/qZ.
        Add (m||bind||κ, r) to L_H.
        Return r.
```

**Listing 5.4:** Game 0: FUO-IND-CCA2 game against $\mathcal{SC}$.

The main insight that is used to prove that $\mathcal{SC}$ is secure in the FUO-IND-CCA2 model, is the fact that the symmetric key $K$ is the output of a random oracle. Because of this, an adversary who does not ask $\kappa^*$ to the random oracle *G_Oracle*, will not learn anything more about which message has been encrypted than a passive attacker against **E**.

**Theorem 5.2.3.** *If an adversary $\mathcal{A}$ neither asks the value $\kappa$ to the random oracle G_Oracle nor a string on the form $*||*||\kappa^*$ to the random oracle H_Oracle, then the advantage of $\mathcal{A}$ in the FUO-IND-CCA2 game against $\mathcal{SC}$ is no greater than that of an adversary $\mathcal{B}$ in the IND-PA game against* **E**. *Concretely, for corresponding values of k and $k_e$,*

$$Adv_{FUO-IND-CCA2}(k, t, q_{sc}, q_{usc}, q_g, q_h) \leq Adv_{IND-PA}(k_e, t_1), \tag{5.9}$$

*where $t_1 = O(t) + t_o$ and $t_o$ is the time required to simulate all queries to the four oracles.*

*Proof.* Consider an adversary $\mathcal{A}$ in a game similar to Game 0 above, but with the alteration that the random oracles *G_Oracle* and *H_Oracle* return $\perp$ whenever they receive the queries $\kappa^*$ and $*||*||\kappa^*$. The following game relates the success rate of $\mathcal{A}$ with that of an adversary $\mathcal{B}$ against the IND-PA security of **E**.

```
B_1:
    I ←R Com(k).
    (x_s, y_s) ←R Key_s(I).
    (x_r, y_r) ←R Key_r(I).
    bind ← y_s||y_r.
    (m_0, m_1, state) ←R A_1(I, y_s, y_r; SC_Oracle, USC_Oracle, G_Oracle, H_Oracle).
    state* ← (I, (x_s, y_s), (x_r, y_r), bind, m_0, m_1, state).
    Return (m_0, m_1, state*).

B_2(c*, state*):
    Parse state*.
    n* ←R Z/qZ.
    κ* ← y_r^{n*} mod p.
    r* ←R Z/qZ.
    s* ← n*/(x_s + r*) mod q.
```

```
    σ* ← (c*, r*, s*).
    b' ← A_2(I, y_s, y_r, state, σ*; SC_Oracle, USC_Oracle, G_Oracle, H_Oracle)
    Return b'.

SC_Oracle(m):
    Return SC(x_s, y_r, m).

USC_Oracle(y'_s, σ):
    Return USC(y'_s, x_r, σ).

G_Oracle(κ):
    If κ = κ*:
        Return ⊥.
    Else if (κ, K) is in the list L_G:
        Return K.
    Else:
        K ←R {0,1}^{k_e}.
        Add (κ, K) to L_G.
        Return K.

H_Oracle(m||bind||κ):
    If κ = κ*:
        Return ⊥.
    Else if (m||bind||κ, r) is in the list L_H:
        Return r.
    Else:
        r ←R Z/qZ.
        Add (m||bind||κ, r) to L_H.
        Return r.
```

**Listing 5.5:** Game 1: IND-PA game against **E**.

In the view of $\mathcal{A}$, nothing has changed from Game 0, unless the event that $\mathcal{A}$ asks the forbidden query to one of the random oracles occurs. However, the only way that information about the hidden bit $b$ can be revealed to $\mathcal{A}$ in Game 1 is through the symmetric ciphertext $c$. Hence, the advantage of $\mathcal{B}$ at distinguishing the bit $b$ is equal to the advantage of $\mathcal{A}$. ■

By Lemma 2.5.2, it follows that the difference between the advantage of $\mathcal{A}$ in Game 0 and the IND-PA game is bounded by the probability that $\mathcal{A}$ in fact *does* ask $\kappa^*$ to one of the random oracles in Game 0. In order to bound the probability that this occurs, an adversary $C$ against the Gap Diffie-Hellman problem is used. The adversary is given as input an instance of the Computational Diffie-Hellman problem, and access to a Decisional Diffie-Hellman oracle. A sequence of FUO-IND-CCA2-games played between $C$ and $\mathcal{A}$ is then constructed. Applying Lemma 2.5.2 repeatedly, each game transition is kept bounded. In the final game, the probability that $\mathcal{A}$ asks $\kappa^*$ to one of the random oracles in that game is shown to be equal to the advantage of $C$. Throughout the remainder of this section, the setting is a Schnorr group specified by the parameters $p$, $q$ and $g$. The algorithm $C$ is attempting to solve a generic

GDH problem instance $\{X = g^x \mod p, Y = g^y \mod p; DDH\_Oracle\}$ from that group.

**Theorem 5.2.4.** *The probability that $\mathcal{A}$ asks $\kappa^*$ to G_Oracle or $*||*||\kappa^*$ to H_Oracle in Game 0 is bounded by the advantage of an adversary $C$ against the Gap Diffie Hellman problem, in the Schnorr group over which $\mathcal{SC}$ is defined. Let $k$ be the security parameter for $\mathcal{SC}$, and let $E_0$ be the event that $\kappa^*$ is asked to either of the random oracles. Then*

$$Pr[E_0] \leq Adv_{GDH,C}(k, t_2, q_{ddh}) + \frac{q_{sc} \cdot (q_{sc} + 2q_g + 2q_h + 1) + 2q_{usc}}{2q}. \tag{5.10}$$

*In the above expression, $\mathcal{A}$ is assumed to run in time $t$ and ask at most $q_{sc}$, $q_{usc}$, $q_g$ and $q_h$ queries to the respective oracles. The GDH-adversary $C$ runs in time $t_2 = O(t) + t_s$, where $t_s$ is the time needed to simulate and reply to oracle queries by $\mathcal{A}$. A total of $q_{ddh} = O(q_{sc} + q_{usc} + q_g + q_h)$ queries are asked to the DDH oracle by $C$.*

*Proof.* The proof proceeds as indicated in the previous paragraph. It is long and therefore divided into several parts, each corresponding to a hop between two games.

**Part 1:** The first section of the proof is used to introduce the adversary $C$.

---

Public information:
  Security parameters $k$, $k_q$, $k_e$.
  $k$–bit prime $p$, $k_q$–bit prime $q$ such that $q|(p-1)$.
  Element $g \in \mathbb{Z}_p$ of order $q$.

$C(X, Y; DDH\_Oracle)$:
  $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
  $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
  $bind \leftarrow y_s||y_r$.
  $(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle, G\_Sim, H\_Sim)$.

  $b \xleftarrow{R} \{0, 1\}$.
  $n^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
  $\kappa^* \leftarrow g^{n^*} \mod p$.
  $r^* \leftarrow H\_Sim(m||bind||\kappa^*)$.
  $s^* \leftarrow n^*/(x_s + r^*) \mod q$.
  $K^* \leftarrow G\_Sim(\kappa^*)$.
  $c^* \leftarrow Encr_{K^*}(m_b)$.
  $\sigma^* \leftarrow (c^*, r^*, s^*)$.

  $b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle, G\_Sim, H\_Sim)$.
  $Z \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
  Return $Z$.

$SC\_Oracle(m)$:
  Return $SC(x_s, y_r, m)$.

*USC_Oracle*$(y'_s, \sigma)$:
    Return $USC(y'_s, x_r, \sigma)$.

*G_Sim*$(\kappa)$:
    If $\kappa = \kappa^*$:
        Return $\perp$.
    Else if $(\kappa, K)$ is in the list $L_G$:
        Return $K$.
    Else:
        $K \xleftarrow{R} \{0,1\}^{k_e}$.
        Add $(\kappa, K)$ to $L_G$.
        Return $K$.

*H_Sim*$(m||bind||\kappa)$:
    If $\kappa = \kappa^*$:
        Return $\perp$.
    Else if $(m||bind||\kappa, r)$ is in the list $L_H$:
        Return $r$.
    Else:
        $r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
        Add $(m||bind||\kappa, r)$ to $L_H$.
        Return $r$.

**Listing 5.6:** Game 2: Introduce $C$.

Let $E_0$ be the probability that $\mathcal{A}$ asks the $\kappa^*$-query to one of the oracles in Game 0, $E_2$ be the probability of the same event in Game 2, and $F_2$ the event that Game 0 and Game 2 behave differently. Since the only thing that is changed in $\mathcal{A}$'s view between the two games is the response when $\kappa^*$ is queried to a random oracle, $Pr[E_0] = Pr[F_2] = Pr[E_2]$. Although $C$ simply returns a random element of $\mathbb{Z}/q\mathbb{Z}$ (and hence has advantage 0), this will change in subsequent games.

**Part 2:** The next step is to replace the signing oracle with a simulator that does not assume knowledge of Alice's private key. To accommodate this, the random oracle simulators will fix the images of certain queries, without knowing the queries themselves. In order to work around this, two additional lists are used by the random oracles to maintain state. The list $L'_G$ contains entries of the form $(\omega, K)$, representing the relation $G\_Sim(\omega^{x_r} \mod p) = K$. This can be tested for in *G_Sim* by checking whether $DDH\_Oracle(y_r, \omega, \kappa) = \top$. Similarly, the list $L'_H$ contains entries of the form $(\omega, m||bind, r)$, and represents the relation $H\_Sim(m||bind||\omega^{x_r} \mod p) = r$.

The following oracle simulators replace those used in Game 2:

*SC_Sim*$(m)$:
    $K \xleftarrow{R} \{0,1\}^{k_e}$.

$c \leftarrow Encr_K(m)$.
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\omega \leftarrow (y_s \cdot g^r)^s \mod p$.
Add $(\omega, K)$ to $L'_G$.
Add $(\omega, m||bind, r)$ to $L'_H$.
$\sigma \leftarrow (c, r, s)$.
Return $\sigma$.

*G_Sim*($\kappa$):
If $\kappa = \kappa^*$:
Return $\perp$.
Else if $(\kappa, K)$ is in the list $L_G$:
Return $K$.
Else if $DDH(y_r, \omega, \kappa) = \top$ for some $(\omega, K)$ in $L'_G$:
Return $K$.
Else:
$K \xleftarrow{R} \{0,1\}^{k_e}$.
Add $(\kappa, K)$ to $L_G$.
Return $K$.

*H_Sim*($m||bind||\kappa$):
If $\kappa = \kappa^*$:
Return $\perp$.
Else if $(m||bind||\kappa, r)$ is in the list $L_H$:
Return $r$.
Else if $(\omega, m||bind, r)$ is in $L'_H$ and $DDH(y_r, \omega, \kappa) = \top$:
Return $r$.
Else:
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
Add $(m||bind||\kappa, r)$ to $L_H$.
Return $r$.

**Listing 5.7:** Game 3: Introduce *SC_Sim*

Let $F_3$ be the event that causes Game 3 and Game 2 to differ, and $E_3$ the probability that $\mathcal{A}$ asks $\kappa^*$ to a random oracle in Game 3. By Lemma 2.5.2, $Pr[E_2] - Pr[E_3 \wedge \neg F_3] \leq Pr[F_3]$. Since the signcryption simulator chooses $K$, $r$ and $s$ uniformly at random and independently, signcryptexts output by *SC_Sim* in Game 3 will have the same distribution as signcryptexts output by the oracle in Game 2. Furthermore, since the images of *G_Sim*($\kappa$) and *H_Sim*($\kappa$) are fixed in $L'_G$ and $L'_H$ by *SC_Sim*, the unsigncryption oracle will not be affected by the change. The only event that will cause Game 3 to differ from Game 2 is when *SC_Sim* adds an implicit query/response pair to $L'_G$ or $L'_H$, and the preimages of $K$ and $r$ have already been fixed by previous queries. Additionally, it is necessary to ensure that $\kappa^*$ is not the correct preimage of these values. This is because $\mathcal{A}$ is explicitly restricted from using *G_Sim* and *H_Sim* to hash $\kappa^*$ directly, and must therefore also be prevented from doing so indirectly through *SC_Sim*.

**Claim 5.2.5.** The probability of $F_3$ occuring in Game 3 is at most $\frac{q_{sc}\cdot(2q_g+2q_h+q_{sc}+1)}{2q}$.

In the worst case, $\mathcal{A}$ asks $q_g$ different queries to *G_Sim* and $q_h$ different queries with identical values for $m$ and *bind* to *H_Sim*, before asking the $q_{sc}$ queries of $m$ to *SC_Sim*. In this case, there are $q_g + q_h + 1$ values that are reserved and causes an error to occur if their images are reassigned in $L'_G$ or $L'_H$ by future queries to *SC_Sim*. Since each query to *SC_Sim* fixes another preimage, the probability that $q_{sc}$ queries complete successfully is at most equal to

$$\sum_{i=1}^{q_{sc}} \frac{q_g + q_h + 1 + (i-1)}{q} = \frac{q_{sc}\cdot(q_g+q_h+1)}{q} + \frac{q_{sc}\cdot(q_{sc}-1)}{2q} = \frac{q_{sc}\cdot(2q_g+2q_h+q_{sc}+1)}{2q}. \quad (5.11)$$

From the above, it follows that

$$Pr[E_0] \leq Pr[E_3 \wedge \neg F_3] + \frac{q_{sc}\cdot(2q_g+2q_h+q_{sc}+1)}{2q}. \quad (5.12)$$

It is worth remarking that this bound differs from that of Baek, Steinfeld and Zheng [BSZ02]. This is due to the fact that they do not consider preimages fixed by subsequent queries to *SC_Sim*, thereby losing the term $\frac{q_{sc}\cdot(q_{sc}-1)}{2q}$. Because different queries to *SC_Sim* may produce identical values of $\omega$ although $K$ and/or $r$ are not the same, this needs to be taken into account.

**Part 3:** In the final transition, the flexible unsigncryption oracle *USC_Oracle* is replaced with a simulation that does not depend on Bob's private receiving key $x_r$. At the same time, the execution of $C$ is altered so that the value $\kappa^*$ corresponds to the correct solution of the Gap Diffie-Hellman problem. The Decisional Diffie-Hellman oracle is used by the random oracle simulators to detect whether $\kappa^*$ is the value being queried. This transition is again based on the probability of a failure event occuring: that of the simulator *USC_Sim* behaving differently from the corresponding oracle in Game 3.

---

$C(X, Y; DDH\_Oracle)$:

$r^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$s^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$y_s \leftarrow (X \cdot g^{-r^*\cdot s^*})^{\frac{1}{s^*}} \mod p$.
$y_r \leftarrow Y$.
$K^* \xleftarrow{R} \{0,1\}^{k_e}$.
$bind^* \leftarrow y_s || y_r$.

Run $\mathcal{A}_1(I, y_s, y_r; SC\_Sim, USC\_Sim, G\_Sim, H\_Sim)$.
If $\mathcal{A}_1$ queries $\kappa$ to *G_Sim* or $m||bind||\kappa$ to *H_Sim* such that $DDH\_Oracle(X, Y, \kappa) = \top$:
  Set $\kappa^* \leftarrow \kappa$.
$\mathcal{A}_1$ returns $(m_0, m_1, state)$.

$b \xleftarrow{R} \{0,1\}$.
$c^* \leftarrow Encr_{K^*}(m_b)$.
$\sigma^* \leftarrow (c^*, r^*, s^*)$.

Run $\mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Sim, USC\_Sim, G\_Sim, H\_Sim)$.
If $\mathcal{A}_2$ queries $\kappa$ to $G\_Sim$ or $m||bind||\kappa$ to $H\_Sim$ such that $DDH\_Oracle(X, Y, \kappa) = \top$:
Set $\kappa^* \leftarrow \kappa$.
$\mathcal{A}_2$ returns $b'$.

If $\kappa^*$ has been set:
Return $\kappa^*$.
Else:
$Z \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
Return $Z$.

$SC\_Sim$ is unchanged from Game 3.

$USC\_Sim(y'_s, \sigma)$:
Parse $(c, r, s) \leftarrow \sigma$.
$\omega \leftarrow (y'_s \cdot g^r)^s \mod p$.
If $\omega = X$:
Return $\perp$.
$bind' \leftarrow y'_s||y_r$.
If $(\omega, K')$ is in $L'_G$ or if $(\kappa, K')$ is in $L_G$ such that $DDH\_Oracle(y_r, \omega, \kappa) = \top$:
$K \leftarrow K'$.
Else:
$K \xleftarrow{R} \{0,1\}^{k_e}$.
Add $(\omega, K)$ to $L'_G$.
$m \leftarrow Decr_K(c)$.
If $(m||bind'||\kappa, \hat{r})$ is in $L_H$ and $DDH\_Oracle(y_r, \omega, \kappa) = \top$ or if $(\omega, m||bind', \hat{r})$ is in $L'_H$:
$r' \leftarrow \hat{r}$.
Else:
$r' \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
Add $(\omega, m||bind, r')$ to $L'_H$.
If $r = r'$:
Return $m$.
Else:
Return $\perp$.

$G\_Sim(\kappa)$:
If $DDH\_Oracle(X, Y, \kappa) = \top$:
$\kappa = \kappa^*$ is the GDH solution.
Return $\perp$.
Else if $(\kappa, K)$ is in the list $L_G$:
Return $K$.
Else if $DDH\_Oracle(y_r, \omega, \kappa) = \top$ for some $(\omega, K)$ in $L'_G$:

```
        Return K.
    Else:
        K ←R {0,1}^{k_e}.
        Add (κ, K) to L_G.
        Return K.

H_Sim(m||bind||κ):
    If DDH_Oracle(X, Y, κ) = ⊤:
        κ = κ* is the GDH solution.
        Return ⊥.
    Else if (m||bind||κ, r) is in the list L_H:
        Return r.
    Else if (ω, m||bind, r) is in L'_H and DDH_Oracle(y_r, ω, κ) = ⊤:
        Return r.
    Else:
        r ←R ℤ/qℤ.
        Add (m||bind||κ, r) to L_H.
        Return r.
```

**Listing 5.8:** Game 4: Introduce *USC_Sim* and modify $C$.

Let $F_4$ be the event that causes the execution of Game 3 and Game 4 to differ, and $E_4$ the event that $\mathcal{A}$ asks $\kappa^*$ (whose value is not known, but detected using *DDH_Oracle*) to one of the random oracles. From the specification of the games, it follows that $Pr[E_3 \wedge \neg F_3 \wedge \neg F_4] = Pr[E_4 \wedge \neg F_3 \wedge \neg F_4]$ and hence the difference $Pr[E_3 \wedge \neg F_3] - Pr[E_4 \wedge \neg F_3 \wedge \neg F_4] \leq Pr[F_4]$ by Lemma 2.5.2.

**Claim 5.2.6.** The probability of $E_4$ occuring is equal to the advantage of $C$ in Game 4.

The GDH-adversary $C$ uses the given GDH problem instance $(X, Y)$ to rig the public keys provided to $\mathcal{A}$ in Game 4. This does not change the the distributions of the input sent to $\mathcal{A}$, but results in the value $\kappa^*$ (which $C$ does not know, but can detect using *DDH_Oracle*) becoming the correct solution to the GDH instance that $C$ is attempting to solve. This means that the advantage of $C$ against the GDH problem is equal to $Pr[E_4]$, since it finds the value of $\kappa^*$ if and only if $\mathcal{A}$ sends it to *G_Sim* or *H_Sim*.

Examining *USC_Sim* closely, one may conclude that there is only one possible way that $F_4$ can occur. Every parameter returned to $\mathcal{A}$ from the simulator is distributed identically to those returned by the oracle. The simulator does not run the risk of adding conflicting entries to $L'_G$ or $L'_H$ the way *SC_Sim* did either, since it only modifies the lists whenever the corresponding entry does not already exist. However, when $\omega = X$, the simulator may differ from the previous oracle. In this case the unsigncryption process can not be completed as usual, since $C$ must not allow $\mathcal{A}$ to gain any information about the images *G_Sim*$(\kappa^*)$ and *H_Sim*$(* || * || \kappa^*)$. If this was allowed to happen, $\mathcal{A}$ would be able to learn some information about the hidden bit $b$ *without* asking $\kappa^*$ to a random oracle simulator, defeating the entire purpose of the exercise.

**Claim 5.2.7.** The probability of *USC_Oracle* causing Game 4 to differ from Game 3, $Pr[F_4]$, is bounded by $\frac{q_{usc}}{q}$.

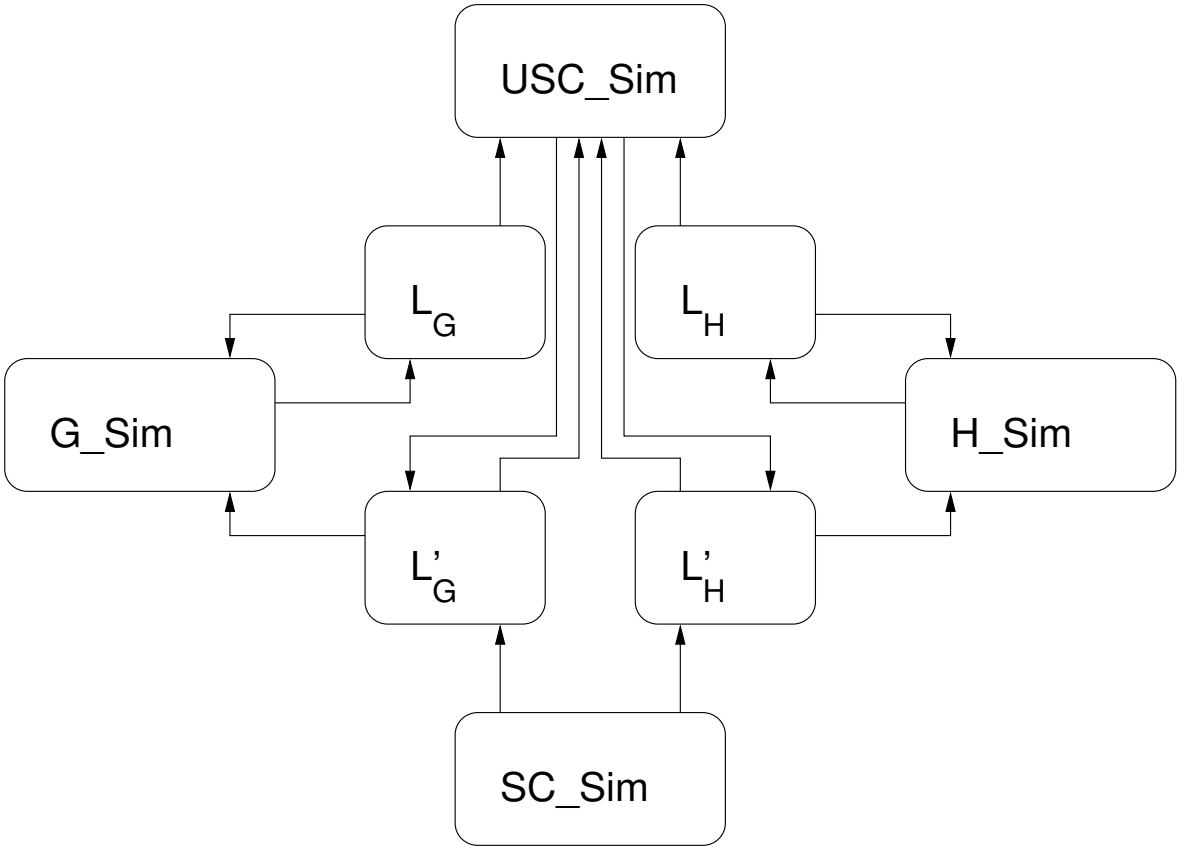



**Figure 5.1:** The information flow between the oracle simulators algorithms in Game 4. All four algorithms add entries to one or two state information lists, and all except *SC_Sim* read data from the lists to maintain consistency.

To see this, consider a specific query to *USC_Sim* that causes $F_4$ to occur, denoted $y_e, (c_e, r_e, s_e)$. It is known that $X = \omega_e = (y_e \cdot g^{r_e})^{s_e} \mod p$. Furthermore, an error will only occur when $r_e = H\_Sim(m_e||y_e||y_r||\kappa^*)$, since the real unsigncryption oracle would also return $\perp$ if this was not the case. Assume (by contradiction) that the string $m_e||y_e||y_r||\kappa^*$, which is the preimage of $r_e$, is equal to the bit-string $m_b||y_s||y_r||\kappa^*$, the preimage of $r^*$. This would imply that $r_e = r^*$. Furthermore, the unsigncryption oracle in Game 4 would compute use the symmetric key $K^*$ from $\kappa$, and decrypt $Decr_{K^*}(c_e) = m_e = m_b = Decr_{K^*}(c^*)$. Hence $c_e = c^*$, since *Decr* is one-to-one. Finally, since $(y_e \cdot g^{r_e})^{s_e} = (y_s \cdot g^{r^*})^{s_e} = X = (y_s \cdot g^{r^*})^{s^*} \mod p$ and all elements of $\mathbb{Z}/q\mathbb{Z}$ are of order $q$ or 1, it follows that $s_e = s^* \mod q$ whenever $X$ is not equal to the identity element, in which case the GDH problem instance is trivial[6]. Hence, if $m_e = m_b$ and $y_e = y_s$, the unsigncryption query is equal to the challenge ciphertext, which is not allowed. The probability that $r_e = H\_Sim(m_e||y_e||y_r||\kappa^*)$ is therefore $\frac{1}{q}$, since the output of the random oracle is uniformly distributed over $\mathbb{Z}/q\mathbb{Z}$ and independent of the input value $r_e$. A final bound is obtained by summing over the $q_{usc}$ oracle queries made by $\mathcal{A}$.

[6]The probability of this occuring is of course completely negligible for a random GDH instance, and may be tested for explicitly by $C$. This is not done because it would obscure the central argument of the proof.

Since $Pr[E_4 \wedge \neg F_3 \wedge \neg F_4] \leq Pr[E_4] = Adv_{GDH,C}(k, t^*, q_{ddh})$ in Game 4, the stated probability bound on $Pr[E_0]$ in Game 0 is obtained by adding up the transitions along the game sequence. ■

**Theorem 5.2.8 (Confidentiality of $\mathcal{SC}$).** *If the Gap Diffie-Hellman problem is hard and the symmetric encryption scheme $\mathbf{E} = \{Encr, Decr\}$ is IND-PA secure, then Zheng's signcryption scheme $\mathcal{SC}$ is secure in the Random Oracle Model with the following bound:*

$$Adv_{FUO-IND-CCA2}(k, t, q_{sc}, q_{usc}, q_g, q_h) \leq$$
$$Adv_{IND-PA}(k_e, t_1) + Adv_{GDH}(k, t_2, q_{ddh}) + \frac{q_{sc} \cdot (q_{sc} + 2q_g + 2q_h + 1) + 2q_{usc}}{2q}. \quad (5.13)$$

*In the above equation, $q_{sc}, q_{usc}, q_g$ and $q_h$ denote the number of oracle queries to the signcryption, flexible unsigncryption and random oracles, while $q_{ddh} = O(q_{sc} + q_g + q_h)$ is the number of queries to the Decisional Diffie-Hellman oracle. The execution times $t_1 = O(t) + t_o$ and $t_2 = O(t) + t_s$, where $t_o$ is the time needed to answer $q_{sc}, q_{usc}, q_g$ and $q_h$ queries to the respective oracles legitimately in Game 0, and $t_s$ is the corresponding time needed to simulate the same number of oracle queries in Game 4 above.*

*Proof.* The stated result follows from Lemma 2.5.2, Theorem 5.2.3 and Theorem 5.2.4. ■

The most interesting thing about this bound is that the lead coefficient of $Adv_{GDH}$ has been halved from the result of Baek, Steinfeld and Zheng [BSZ02] as stated in Claim 5.2.1. This may possibly be attributed to the structured game hopping approach used in the proof, permitting a clearer view of how the different probability bounds relate to each other. The additional terms relating to the number of oracle queries stem from the transition from Game 2 to Game 3, where *SC_Sim* is introduced; as previously commented, the original paper does not take into account that successive queries to *SC_Sim* increase the number of reserved preimages in the oracle simulators.

# 6

# Security of Zheng's signcryption scheme in the hybrid model

This chapter examines the security of Zheng's signcryption scheme in the hybrid security model introduced by Dent [Den04], and compares the results to those of the previous chapter.

## 6.1 A hybrid formulation of Zheng's signcryption scheme

As seen in Chapter 3.3, Zheng's signcryption scheme combines the features of a hybrid encryption scheme and a signature schemes in an elegant fashion. In Chapter 4.2, the hybrid security model for signcryption schemes proposed by Dent [Den04] is discussed. It is therefore very natural to examine the security of Zheng's scheme in the hybrid framework, and compare the results with those obtained under the more traditional security models used in Chapter 5.

Zheng's signcryption scheme $\mathcal{SC}$ may be constructed from the corresponding signcryption KEM **SC_KEM** listed below, together with a signcryption DEM **SC_DEM** whose decryption function is one-to-one and whose symmetric keys are of the same length as the ones output by **SC_KEM**.

**Definition 6.1.1 (Zheng's signcryption KEM).** Zheng's signcryption KEM **SC_KEM** = $\{Com, Key_s, Key_r, Encap, Decap, Ver\}$ consists of six algorithms. Let $k \in \mathbb{N}$ be an overall security parameter, and $k_q, k_e \in N$ additional security parameters that are derived from $k$. The algorithms are then specified as follows:

---

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 $g \in \mathbb{Z}/q\mathbb{Z}$ of order $q$.
  Choose a cryptographic hash function $\mathcal{G} : \{0,1\}^* \to \{0,1\}^{k_e}$.
  Choose a cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}$.
  Return $I \leftarrow (p, q, g, \mathcal{G}, \mathcal{H})$.

$Key_s(I)$:
  $x_s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.

$y_s \leftarrow g^{x_s} \mod p$.
Return $(x_s, y_s)$.

$Key_r(I)$:
$x_r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$y_r \leftarrow g^{x_r} \mod p$.
Return $(x_r, y_r)$.

$Encap(x_s, y_r, m)$:
$n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa \leftarrow y_r^n \mod p$.
$bind \leftarrow y_s || y_r$.
$r \leftarrow \mathcal{H}(m||bind||\kappa)$.
$s \leftarrow n/(x_s + r) \mod q$.
$K \leftarrow \mathcal{G}(\kappa)$.
$e \leftarrow (r, s)$.
Return $\sigma \leftarrow (K, e)$.

$Decap(y_s, x_r, e)$:
Parse $(r, s) \leftarrow e$.
$\kappa \leftarrow (y_s \cdot g^r)^{x_r \cdot s} \mod p$.
$K \leftarrow \mathcal{G}(\kappa)$.
Return $K$.

$Ver(y_s, x_r, m, e)$:
Parse $(r, s) \leftarrow e$.
$bind \leftarrow y_s || y_r$.
$\kappa \leftarrow (y_s \cdot g^r)^{x_r \cdot s} \mod p$.
$r' \leftarrow H(m||bind||\kappa)$.
If $r = r'$:
    Return $\top$.
Else:
    Return $\bot$.

**Listing 6.1:** Zheng's signcryption KEM.

Zheng's signcryption KEM is cited as an example of an insider-secure signcryption KEM in [Den04, Section 5.6], with reference to [BSZ02].

## 6.2 Authenticity of Zheng's signcryption KEM

From Theorem 4.2.4, it is known that the signcryption KEM **SC_KEM** as stated above is INT-CCA2 secure if and only if Zheng's signcryption scheme $\mathcal{SC}$ is sUF-CMA secure. However, in Chapter 5.1 it was only shown that Zheng's signcryption scheme is UF-CMA secure. An additional argument is therefore needed to establish that **SC_KEM** in fact does maintain the integrity and authenticity of encapsulations.

**Lemma 6.2.1.** *Zheng's signcryption scheme $\mathcal{SC}$ is sUF-CMA secure if the SDSS1 signature scheme is sUF-CMA secure.*

*Proof.* In the proof of Theorem 5.1.2, an adversary that uses an UF-CMA adversary against $\mathcal{SC}$ to win the UF-CMA game against the SDSS1 signature scheme is constructed. The reduction remains valid if one changes both security models from UF to sUF. When $\mathcal{B}$ receives a legitimate forgery $(c, r, s)$ from $\mathcal{A}$ that unsigncrypts to a message $m \neq \perp$, then $m, (r, s)$ is also a valid SDSS1 forgery. Furthermore, $(r, s)$ will only have been a reply to a query $m$ to $\mathcal{B}$'s signing oracle if $(c, r, s)$ was a reply to a signcryption oracle query $m$ from $\mathcal{A}$, as long as the symmetric decryption function *Decr* is one-to-one. Hence, an advantage bound for the sUF-CMA reduction is identical to the UF-CMA case covered in Theorem 5.1.2. ∎

From the previous lemma, one may conclude that **SC_KEM** is INT-CCA2 secure if SDSS1 signatures are sUF-CMA. It is also apparent that the notions of regular and strong existential unforgeability are not applicable to identification schemes such as SDSS1-ID, since the situation only arises in situations where an adversary may use a signing (or signcryption) oracle. However, existing literature on the unforgeability of signature schemes, such as the forking lemma used by Pointcheval and Stern and the ID Reduction Lemma described in Chapter 5.1 [PS96] [OO98], relies on a tacit assumption that the security model under consideration is UF-CMA, and not sUF-CMA. An argument must therefore be made that SDSS1 signatures are secure in the sUF-CMA model as well. From Lemma 5.1.5 and Theorem 5.1.10, it can be assumed that SDSS1 is UF-CMA secure (relative to the discrete logarithm problem).

Assume therefore that there exists an adversary $\mathcal{A}$ who has access to $q_s$ adaptively chosen message/signature-pairs $(m_i, \sigma_i)$ that are valid under Alice's public key $y_s$. Since UF-CMA security is assumed, $\mathcal{A}$ is unable to create a new valid message/signature-pair $(m, \sigma)$, where $m \neq m_i$ for all $1 \leq i \leq q_s$. If $\mathcal{A}$ is able to forge messages in the sUF-CMA model, then $\mathcal{A}$ must be able to find a new valid signature $\sigma$ on one of the messages $m_i$. To do so, it must use the information contained in the obtained signatures in some way. In the SDSS1 scheme, each message $m_i$ is only used as input to $\mathcal{H}$, which is modelled as a random oracle. This means that the output of $\mathcal{H}$ is completely independent of the message being signed. This fact will be used repeatedly to show that it is no easier for $\mathcal{A}$ to create a valid signature for a message that has been queried to the signing oracle, than for one that has not.

**Proposition 6.2.2.** *If SDSS1 is UF-CMA secure, the probability that an adversary $\mathcal{A}$ asking $q_s$ queries to the signing oracle will be able to produce a valid sUF-CMA forgery $(m_i, \sigma)$ of SDSS1 is negligible. Concretely, the advantage of a sUF-CMA adversary running in time $t$ and asking $q_{sc}$ and $q_h$ queries to its signature oracle is limited by the following bound:*

$$Adv_{sUF-CMA}(k, t, q_s, q_h) \leq Adv_{UF-CMA}(k, t, q_s, q_h) + \frac{q_s \cdot (q_s - 1)}{q}. \tag{6.1}$$

*Proof.* Consider an efficient adversary $\mathcal{A}$ that has asked $q_s$ signing oracle queries to obtain a list of $q_s$ valid message/signature pairs $(m_i, \sigma_i)$. Two separate cases must be considered. Either the signature returned by $\mathcal{A}$ is equal to a $\sigma_j$ previously returned by the signing oracle, or it is a completely new signature on $m_i$.

**Case 1:** In the former case, there are at most $q_s \cdot (q_s - 1)$ possible message/signature pairs that are not query/response pairs from the signing oracle, and thus possible sUF-CMA forgeries.

This occurs whenever $m_i \neq m_j$ and $s_i \neq s_j$ for all $1 \leq i \neq j \leq q_s$. However, the probability that $\top \leftarrow Ver(y_s, m_i, \sigma_j)$ is completely independent of the fact that $(m_i, \sigma_i)$ and $(m_j, \sigma_j)$ are valid signatures. To see this, consider how the verification algorithm functions: $\sigma$ is parsed to yield the pair $(r, s)$, $\kappa$ is computed as $(y_s \cdot g^r)^s \mod p$, and $\mathcal{H}(m\|\kappa)$ is compared to $r$. From the random oracle property, it follows that

$$Pr\Big[\mathcal{H}(m_i\|\kappa_j) = r_j \,|\, \mathcal{H}(m_j\|\kappa_j) = r_j\Big] = Pr\Big[\mathcal{H}(m_i\|\kappa_j) = r_j\Big] = \frac{1}{q}. \tag{6.2}$$

Hence, the probability of finding a valid forgery is in this case at most $\frac{q_s \cdot (q_s - 1)}{q}$.

**Case 2:** If $\mathcal{A}$ is to return a signature on $m_i$ that has not been returned from the signature oracle, it is either a completely new signature $\sigma = (r, s)$, or derived from previously returned signatures. In the first case, finding a pair $(r, s)$ such that $\mathcal{H}\Big(m_i\|(y_s \cdot g^r)^s \mod p\Big) = r$ is no easier for $m_i$ than for any other $m$, and hence not feasible since SDSS1 is assumed to be UF-CMA secure. Otherwise, the signature is on the form $(r_j, s)$ or $(r, s_j)$ for $1 \leq j \leq q_s$, with the restrictions that $s \neq s_j$ and $r \neq r_j$.

If $\sigma = (r_j, s)$, then it is known that $\mathcal{H}(m_j\|\kappa_j) = r_j$. However, any modification of $s$ leads to $(y_s \cdot g^{r_j})^s \neq \kappa_j$ unless $y_s \cdot g^{r_j} = 1$. This will never occur in a signature returned by a honest signing oracle, since it implies that $x_s + r_j = 0 \pmod q$, which causes the signature algorithm to fail. Hence, the probability that $\mathcal{H}\Big(m_i\|(y_s \cdot g^{r_j})^s\Big) = r_j$ for an arbitrary $s \neq s_j$ is equal to $\frac{1}{q}$, even in the case that $i = j$. Since the output from the random oracle is independent of its input, finding a value of $s$ that causes the relation to hold is no easier for $m_i$ than for an arbitrary $m$. Again, this leads to the conclusion that no efficient algorithm exists to do this, by the assumption that SDSS1 is UF-CMA secure.

The final possibility to be considered is that $\mathcal{A}$ returns a signature on the form $(r, s_j)$. This implies that $\mathcal{H}\Big(m_i\|(y_s \cdot g^r)^{s_j} \mod p\Big) = r$. The probability that this is the case for an arbitrary $r$ is $\frac{1}{q}$ by the random oracle property, and completely independent of $m_i$. Since the adversary has a negligible advantage at finding such an $r$ for any $m$ that has *not* been asked to the signing oracle, it does not have any greater advantage at finding it for $m_i$. ∎

**Theorem 6.2.3 (INT-CCA2 security of SC_KEM).** *The advantage function for the problem of breaking the INT-CCA2 security of* **SC_KEM** *with security parameter k in time t using $q_{enc}, q_h$ and $q_g$ queries to the encapsulation oracle and random oracles, is bounded as follows:*

$$Adv_{INT-CCA2}(k, t, q_{enc}, q_g, q_h) \leq Adv_{UF-CMA}(k, t', q_{enc}, q_{enc} + q_h) + \frac{2q_h + 3q_{enc} \cdot (q_{enc} - 1)}{2q}. \tag{6.3}$$

*In the above equation, the UF-CMA advantage with respect to SDSS1 is taken over adversaries with running time $t' = O(t) + t_1 + t_2$ asking $q_{enc}$ queries to its signature oracle and $q_{enc} + q_h$ queries to its random oracle. The time $t_1$ is the overhead incurred in the reductions from* **SC_KEM** *to $\mathcal{SC}$, and is dominated by the time needed to simulate $q_{enc}$ encapsulation oracle queries. Similarly, $t_2$ denotes the overhead in the reduction from $\mathcal{SC}$ to SDSS1, and consists mainly of the time used to simulate of $q_{enc}$ signcryption oracle queries and $q_h$ random oracle queries.*

*Proof.* The stated result follows from Theorem 4.2.4, Theorem 5.1.2, Lemma 6.2.1 and Proposition 6.2.2. ∎

## 6.3 Confidentiality of Zheng's signcryption KEM

The confidentiality of Zheng's signcryption KEM is captured by the notions of IND-CCA2 and INP-CCA2. It is not surprising that the security of **SC_KEM** under these notions can be proved in a similar fashion to the proof that $\mathcal{SC}$ is FUO-IND-CCA2-secure in Chapter 5.2.

---

Public information:
Security parameters $k$, $k_q$.

IND–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$bind \leftarrow y_s || y_r$.
$(m, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r;$ *Encap_Oracle*, *Decap_Oracle*, *Ver_Oracle*, *G_Oracle*, *H_Oracle*$)$.

$b \xleftarrow{R} \{0,1\}$.
$n^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa^* \leftarrow y_r^{n^*} \mod p$.
$K_0 \leftarrow \mathcal{G}(\kappa^*)$.
$r^* \leftarrow \mathcal{H}(m||bind||\kappa^*)$.
$s^* \leftarrow n^*/(x_s + r^*) \mod q$.
$e^* \leftarrow (r^*, s^*)$.
$K_1 \xleftarrow{R} \{0,1\}^{k_e}$.

$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, K_b, e^*;$ *Encap_Oracle*, *Decap_Oracle*, *Ver_Oracle*, *G_Oracle*, *H_Oracle*$)$.
$\mathcal{A}$ wins if $b' = b$ and $\mathcal{A}_2$ has not asked $e^*$ to *Decap_Oracle*.

INP–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r;$ *Encap_Oracle*, *Decap_Oracle*, *Ver_Oracle*, *G_Oracle*, *H_Oracle*$)$.

$b \xleftarrow{R} \{0,1\}$.
$n^* \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa^* \leftarrow y_r^{n^*} \mod p$.
$r^* \leftarrow \mathcal{H}(m_b||bind||\kappa^*)$.
$s^* \leftarrow n^*/(x_s + r^*) \mod q$.

$e^* \leftarrow (r^*, s^*)$.

$b' \overset{R}{\leftarrow} \mathcal{A}_2(I, y_s, y_r, state, e^*;$ *Encap_Oracle*, *Decap_Oracle*, *Ver_Oracle*, *G_Oracle*, *H_Oracle*).
$\mathcal{A}$ wins if $b' = b$ and $\mathcal{A}_2$ has not asked $(m_0, e^*)$ or $(m_1, e^*)$ to *Ver_Oracle*.

*Encap_Oracle*($s$):
Return *Encap*$(x_s, y_r, m)$.

*Decap_Oracle*($e$):
Return *Decap*$(y_s, x_r, e)$.

*Ver_Oracle*($m, e$):
Return *Ver*$(y_s, x_r, m, e)$.

*G_Oracle*($\kappa$):
If $(\kappa, K)$ is in the list $L_G$:
Return $K$.
Else:
$K \overset{R}{\leftarrow} \{0,1\}^{k_e}$.
Add $(\kappa, K)$ to $L_G$.
Return $K$.

*H_Oracle*($m||bind||\kappa$):
If $(m||bind||\kappa, r)$ is in the list $L_H$:
Return $r$.
Else:
$r \overset{R}{\leftarrow} \mathbb{Z}/q\mathbb{Z}$.
Add $(m||bind||\kappa, r)$ to $L_H$.
Return $r$.

**Listing 6.2:** IND-CCA2 and INP-CCA2 games against **SC_KEM**.

The key concept needed to bound the advantage of $\mathcal{A}$ in either attack game, is again that $\mathcal{A}$ needs to recover $\kappa^*$ and use the the random oracles to reveal any information about $b$. In the IND-CCA2 game the real key $K_0 \leftarrow \mathcal{G}(\kappa)$ is completely indistinguishable from the random key $K_1$, since it is output by a random oracle. Similarly, the only place $m_b$ is used in the INP-CCA2 game is as input to the random oracle $\mathcal{H}$. Hence, to bound the advantage of an adversary winning either game, it is sufficient to bound the probability that said adversary asks $\kappa$ to either random oracle. This is precisely what was done in the proof of Theorem 5.2.4. In particular, one may notice that the technique used to prove Theorem 5.2.4 does not concern itself with what sort of game $\mathcal{A}$ is trying to win. Instead, it examines the probability that $\mathcal{A}$ asks $\kappa^*$ to a random oracle when running with the given computational resources. In the IND-CCA2 and INP-CCA2 games it is not even necessary to concern oneself with the symmetric cipher used, since that is taken care of by **SC_DEM**.

**Proposition 6.3.1.** *Zheng's signcryption KEM* **SC_KEM** *is IND-CCA2 and INP-CCA2 secure*

*relative to the Gap Diffie-Hellman problem under the Random Oracle model, with the following bound:*

$$Adv_{IND-CCA2}(k,t,q_{enc},q_{dec},q_{ver},q_g,q_h) = Adv_{INP-CCA2}(k,t,q_{enc},q_{dec},q_{ver},q_g,q_h) \leq Adv_{GDH}(k,t',q_{ddh}) + \frac{q_{enc}\cdot(q_{enc}+2q_g+2q_h+1)+2q_{dec}}{2q}. \quad (6.4)$$

*In the above expression, $t' = O(t) + t_o$, where $t_o$ is the time required to simulate answers to all oracle queries. The number of queries asked to the Decisional Diffie-Hellman oracle is given by $q_{ddh} = O(q_{enc} + q_{ver} + q_g + q_h)$.*

*Proof.* The proof is omitted. It is identical to the proof of Theorem 5.2.4, with the obvious modifications discussed above. ■

Direct application of Theorem 4.2.5 yields a bound for the security of $\mathcal{SC}$ in the IND-CCA2 security model, when viewed as a hybrid signcryption scheme.

**Theorem 6.3.2.** *If the Gap Diffie-Hellman problem is hard and the decryption algorithm of* **SC_DEM** *is one-to-one, then Zheng's signcryption scheme $\mathcal{SC}$ is secure in the Random Oracle Model with the following bound:*

$$Adv_{IND-CCA2}(k,t,q_{sc},q_{usc},q_g,q_h) \leq 4\cdot Adv_{GDH}(k,t_1,q_{ddh}) + Adv_{INT-CCA2}(k,t_2,q_e) + Adv_{IND-PA}(k,t_3) + \frac{2q_{enc}\cdot(q_{enc}+2q_g+2q_h+1)+2q_{dec}}{q}. \quad (6.5)$$

*The times $t_1$, $t_2$ and $t_3$ corresponding to the times needed by the different adversaries to run $\mathcal{A}$ and simulate the appropriate number of oracle queries, as discussed in Theorem 4.2.5 and 5.2.4.*

This is very similar to the bound reached in by Baek, Steinfeld and Zheng cited in Claim 5.2.1, as well as the obtained result stated in Theorem 5.2.5. Apart from a constant multiplicative factor, the main difference to be found is the additional term corresponding to the INT-CCA2 security of **SC_KEM**.

## 6.4 Problems with Dent's hybrid model

Dent's framework for hybrid signcryption is very appealing to work with, because it cleanly separates the notion of confidential and authenticated key transfer from the security of the symmetric encryption used for the message payload itself. However, there are certain weaknesses that are illustrated by the security reductions for $\mathcal{SC}$, relative to the reductions performed in Chapter 5 and by Baek, Steinfeld and Zheng under more conventional security models.

The unforgeability bound obtained for $\mathcal{SC}$ relative to **SC_KEM** is very nice, as shown in Theorem 4.2.4. The confidentiality of hybrid signcryption causes greater problems, as illustrated by the results in Chapter 6.2. It is not surprising that a generic reduction from the security of **SC_KEM** and **SC_DEM** will yield a somewhat looser bound for the confidentiality of $\mathcal{SC}$ than the direct proof performed in Chapter 5.2. As such, the appearance of a constant factor 4 in front of some terms is not unexpected. What is surprising and also somewhat problematic, is the fact that the INT-CCA2 bound for **SC_KEM** appears in the reduction. This is surprising because unforgeability and confidentiality are usually seen as entirely separate security

goals. Even though signcryption as such desires to obtain both, it is not expected that one relies on the other. The problematic aspect of this is the effect that this additional term makes on the efficiency of the reduction. Although the unforgeability term by itself is negligible, there is a clear effect when examining the *concrete security* [Bel99] of $\mathcal{SC}$. As seen in Chapter 5.1, the unforgeability of $\mathcal{SC}$ is quadratic in its reduction to the discrete logarithm problem, whereas the confidentiality of $\mathcal{SC}$ has a linear reduction to the Gap Diffie-Hellman problem. Under the original reduction, security parameters for practical usage can thus be chosen with respect to the desirable hardness of either problem. In Dent's framework for hybrid signcryption, the confidentiality of $\mathcal{SC}$ hinges on both problems, and security parameters must be chosen with respect to their total hardness. Koblitz and Menezes [KM04] argue that the quadratic security reductions based on the splitting lemma are so weak that they yield no practical security guarantees unless the security parameters used are prohibitively large.

Another complication arising in Dent's model is that it requires the sUF-CMA security of **SC_KEM**, rather than the more well-studied UF-CMA problem. Until the difference between the notions strong and regular existential unforgeability has been studied further, it remains a problem that it may often be difficult to construct an argument that a given scheme is strongly unforgeable. This is in sharp contrast to the regular unforgeability notion, for which standard reduction techniques often may be applied. Furthermore, any uncertainty about the strong unforgeability of a scheme immediately causes an otherwise solid confidentiality statement to collapse in the hybrid model. The original reduction [BSZ02] demonstrates that $\mathcal{SC}$ does not need to be secure against existential forgery to be secure with respect to indistinguishability of signcryptions. It is therefore very interesting to see whether it is possible to maintain this property when analysing arbitrary hybrid signcryption schemes in Dent's framework.

## 6.5 Possible modifications to Dent's hybrid model

As discussed in the previous section, one would like to lose the term depending on the INT-CCA2 security of the **SC_KEM** from the bound for the IND-CCA2 security of a hybrid signcryption scheme **SC**. In this section an alternate reduction will be attempted, to examine what tradeoffs will have to be made if it is left out of the reduction. The following theorem states an alternate reduction for the IND-CCA2 security of a hybrid signcryption scheme **SC**.

**Theorem 6.5.1 (Alternate confidentiality bound for hybrid signcryption).** *Let* **SC** = $\{Gen, Key_s, Key_r, SC, USC\}$ *be a hybrid signcryption scheme constructed from* **SC_KEM** = $\{Gen, Key_s, Key_r, Encap, Decap\}$ *and* **SC_DEM** = $\{Encr, Decr\}$ *whose decryption algorithm is one-to-one. Suppose that there exists an adversary $\mathcal{A}$ that wins the IND-CCA2 game for* **SC**, *with running time $t$, that asks $q_{sc}$ queries to its signcryption oracle and $q_{usc}$ queries to its unsigncryption oracle. Then there exists adversaries $\mathcal{B}$ and $C$, against* **SC_KEM** *and $\mathcal{D}$ against* **SC_DEM**, *such that*

$$Adv_{IND-CCA2,\mathcal{A}}(k, t, q_{sc}, q_{usc}) \leq 2 \cdot Adv_{IND-CCA2,\mathcal{B}}(k, t_1, q_{enc}, q_{dec}, q_{dec}) + Adv_{INP-CCA2,C}(k, t_2, q_{enc}, q_{dec}, q_{dec}) + Adv_{IND-CCA2,\mathcal{D}}(k, t_3, q_{decr}). \quad (6.6)$$

*The running times and number of oracle queries used by the different adversaries are identical to the corresponding adversaries in Theorem 4.2.5, with the exception that $\mathcal{D}$ is allowed to ask $q_{decr} \leq q_{usc}$ queries to an decryption oracle for the* **SC_DEM**.

*Proof.* The proof closely follows the proof of [Den04, Thm. 42]. It differs from the original in that one game is omitted; this also makes a difference in the final game.

**Part 1:** Let Game 0 be the original IND-CCA2 game against **SC**.

---

Public information:
Security parameters $k$, $k_e$.

IND–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$bind \leftarrow y_s \| y_r$.
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$.

$b \xleftarrow{R} \{0,1\}$.
$(K^*, e^*) \xleftarrow{R} Encap(x_s, y_r, m_b)$.
$c^* \leftarrow Encr_{K^*}(m_b)$.
$\sigma^* \leftarrow (c^*, e^*)$.

$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$.
$\mathcal{A}$ wins if $b = b'$ and $\sigma^*$ has not been asked to $USC\_Oracle$.

$SC\_Oracle(m)$:
Return $SC(x_s, y_r, m)$.

$USC\_Oracle(\sigma)$:
Return $USC(y_s, x_r, \sigma)$.

---

**Listing 6.3:** Game 0: IND-CCA2 game against **SC**.

The following game is a modification of Game 0, where a random key is used to encrypt the challenge signcryptext, rather than the one output by $Encap(x_s, y_r, m_b)$.

---

IND–CCA2–experiment:
$I \xleftarrow{R} Com(k)$.
$(x_s, y_s) \xleftarrow{R} Key_s(I)$.
$(x_r, y_r) \xleftarrow{R} Key_r(I)$.
$bind \leftarrow y_s \| y_r$.
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$.

$b \xleftarrow{R} \{0,1\}$.
$(K^*, e^*) \xleftarrow{R} Encap(x_s, y_r, m_b)$.

$K' \xleftarrow{R} \{0,1\}^{k_e}$.
$c^* \leftarrow Encr_{K'}(m_b)$.
$\sigma^* \leftarrow (c^*, e^*)$.

$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$.
$\mathcal{A}$ wins if $b = b'$ and $\sigma^*$ has not been asked to $USC\_Oracle$.

$SC\_Oracle(m)$:
Return $SC(x_s, y_r, m)$.

$USC\_Oracle(\sigma)$:
Return $USC(y_s, x_r, \sigma)$.

**Listing 6.4:** Game 1: Modified IND-CCA2 game against **SC**.

**Claim 6.5.2.** The difference between the advantages of $\mathcal{A}$ in Game 0 and Game 1, is bounded by twice the advantage of an adversary $\mathcal{B}$ against the IND-CCA2 security of **SC_KEM**.

The transition between Game 0 and Game 1 is based on indistinguishability. The following listing gives a concrete specification of the adversary $\mathcal{B}$:

$\mathcal{B}_1(I, y_s, y_r; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$:
$(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$.
$b \xleftarrow{R} \{0,1\}$.
$state^* \leftarrow \left(b, m_0, m_1, state\right)$.
Return $(m_b, state^*)$.

$\mathcal{B}_2(I, y_s, y_r, state^*, K^*, e^*; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$:
Parse $state^*$.
$c^* \xleftarrow{R} Encr_{K^*}(m_b)$.
$\sigma^* \leftarrow (c^*, e^*)$.
$b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$.
If $b = b'$:
    Return 0.
Else:
    Return 1.

$SC\_Oracle(m)$:
$(K, e) \xleftarrow{R} Encap\_Oracle(m)$.
$c \leftarrow Encr_K(m)$.
$\sigma \leftarrow (c, e)$.
Return $\sigma$.

$USC\_Oracle(\sigma)$:
Parse $(c, e) \leftarrow \sigma$.

If $e = e^*$:
    $K \leftarrow K^*$.
Else:
    $K \leftarrow Decap\_Oracle(e)$.
$m \leftarrow Decr_K(c)$.
If $Ver\_Oracle(m, e) = \top$:
    Return $m$.
Else:
    Return $\perp$.

**Listing 6.5:** Distinguisher algorithm $\mathcal{B}$.

The adversary $\mathcal{B}$ plays either Game 0 or Game 1, depending on the hidden bit in the IND-CCA2 game against **SC_KEM**. It then uses $\mathcal{A}$ to distinguish between the games. By application of Lemma 2.5.3, the claimed result follows.

**Part 2:** In Game 2, the message used to create the encapsulation used in the challenge ciphertext is fixed as $m_0$, rather than chosen at random.

IND–CCA2–experiment:
    $I \xleftarrow{R} Com(k)$.
    $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
    $(x_r, y_r) \xleftarrow{R} Key_r(I)$.
    $bind \leftarrow y_s \| y_r$.
    $(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$.

    $b \xleftarrow{R} \{0,1\}$.
    $(K^*, e^*) \xleftarrow{R} Encap(x_s, y_r, m_0)$.
    $K' \xleftarrow{R} \{0,1\}^{k_e}$.
    $c^* \leftarrow Encr_{K'}(m_b)$.
    $\sigma^* \leftarrow (c^*, e^*)$.

    $b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$.
    $\mathcal{A}$ wins if $b = b'$ and $\sigma^*$ has not been asked to $USC\_Oracle$.

$SC\_Oracle(m)$:
    Return $SC(x_s, y_r, m)$.

$USC\_Oracle(\sigma)$:
    Return $USC(y_s, x_r, \sigma)$.

**Listing 6.6:** Game 2: Modified IND-CCA2 game against **SC**.

**Claim 6.5.3.** The difference between the advantages of $\mathcal{A}$ in Game 1 and Game 2, is bounded by the advantage of an adversary $C$ against the INP-CCA2 security of **SC_KEM**.

The transition between Game 1 and Game 2 is again based on indistinguishability. Whenever the bit $b$ is equal to 0, the two games are identical. The following listing specifies an adversary $C$ playing the INP-CCA2 game against **SC_KEM**.

---

$C_1(I, y_s, y_r; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$ :
    $(m_0, m_1, state) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; SC\_Oracle, USC\_Oracle)$ .
    $state^* \leftarrow \big(m_0, m_1, state\big)$ .
    Return $(m_0, m_1, state^*)$ .

$C_2(I, y_s, y_r, state^*, e^*; Encap\_Oracle, Decap\_Oracle, Ver\_Oracle)$ :
    Parse $state^*$ .
    $K^* \xleftarrow{R} \{0,1\}^{k_e}$ .
    $c^* \xleftarrow{R} Encr_{K^*}(m_b)$ .
    $\sigma^* \leftarrow (c^*, e^*)$ .
    $b' \xleftarrow{R} \mathcal{A}_2(I, y_s, y_r, state, \sigma^*; SC\_Oracle, USC\_Oracle)$ .
    Return $b'$ .

$SC\_Oracle(m)$ :
    $(K, e) \xleftarrow{R} Encap\_Oracle(m)$ .
    $c \leftarrow Encr_K(m)$ .
    $\sigma \leftarrow (c, e)$ .
    Return $\sigma$ .

$USC\_Oracle(\sigma)$ :
    Parse $(c, e) \leftarrow \sigma$ .
    If $e = e^*$ :
        $K \leftarrow K^*$ .
    Else :
        $K \leftarrow Decap\_Oracle(e)$ .
    $m \leftarrow Decr_{K^*}(c)$ .
    If $\top \leftarrow Ver\_Oracle(m, e)$ :
        Return $m$ .
    Else :
        Return $\perp$ .

---

**Listing 6.7:** Distinguisher algorithm $C$.

The algorithm $C$ plays the INP-CCA2 game against **SC_KEM**. From the way that it is constructed, whether Game 1 or Game 2 is presented to $\mathcal{A}$ depends on the value of the hidden bit that $C$ is trying to guess. If there is a non-negligible difference in the advantage of $\mathcal{A}$ in Game 1 and Game 2, then $C$ will be able to use this to distinguish which message was encapsulated. The advantage of $C$ at guessing the correct bit is therefore equal to that of $\mathcal{A}$[1].

[1] In [Den04], the corresponding game transition is not bounded explicitly. Instead it is claimed that it may be done using the Distinguisher Lemma (Lemma 2.5.3) in a similar way to the transition between Game 0 and 1. As can be seen from the above construction, a bound may be obtained without using this lemma, thus shaving a

**Part 3:** To obtain a final bound for the advantage of $\mathcal{A}$ in Game 2, an IND-CCA2 adversary against the **SC_DEM** is used.

**Claim 6.5.4.** The advantage of $\mathcal{A}$ in Game 2 is bounded by that of an IND-CCA2 adversary $\mathcal{D}$ against **SC_DEM**.

The following listing specifies the adversary $\mathcal{D}$. It simulates the runtime environment of $\mathcal{A}$ in Game 2 perfectly, and uses $\mathcal{A}$ to determine the encrypted message, winning its game whenever $\mathcal{A}$ does.

---

$\mathcal{D}_1(; \mathit{Decr\_Oracle})$ :
  $I \xleftarrow{R} \mathit{Com}(k)$ .
  $(x_s, y_s) \xleftarrow{R} \mathit{Key}_s(I)$ .
  $(x_r, y_r) \xleftarrow{R} \mathit{Key}_r(I)$ .
  $\mathit{bind} \leftarrow y_s \| y_r$ .
  $(m_0, m_1, \mathit{state}) \xleftarrow{R} \mathcal{A}_1(I, y_s, y_r; \mathit{SC\_Oracle}, \mathit{USC\_Oracle})$ .
  $\mathit{state}^* \leftarrow \big(I, (x_s, y_s), (x_r, y_r), \mathit{bind}, m_0, m_1, \mathit{state}\big)$ .
  Return $(m_0, m_1, \mathit{state}^*)$ .

$\mathcal{D}_2(c^*, \mathit{state}^*; \mathit{Decr\_Oracle})$ :
  Parse $\mathit{state}^*$ .
  $(K^*, e^*) \xleftarrow{R} \mathit{Encap}(x_s, y_r, m_0)$ .
  $\sigma^* \leftarrow (c^*, e^*)$ .
  $b' \leftarrow \mathcal{A}_2(I, y_s, y_r, \mathit{state}, \sigma^*; \mathit{SC\_Oracle}, \mathit{USC\_Oracle})$ .
  Return $b'$ .

$\mathit{SC\_Oracle}(m)$ :
  $(K, e) \xleftarrow{R} \mathit{Encap}(x_s, y_r, m)$ .
  $c \leftarrow \mathit{Encr}_K(m)$ .
  Return $(c, e)$ .

$\mathit{USC\_Oracle}(\sigma)$ :
  Parse $(c, e) \leftarrow \sigma$ .
  If $e = e^*$ :
    $m \leftarrow \mathit{Decr\_Oracle}(c)$ .
  Else :
    $K \leftarrow \mathit{Decap}(y_s, x_r, e)$ .
    $m \leftarrow \mathit{Decr}_K(c)$ .
  If $\mathit{Ver}(y_s, x_r, m, e) = \top$ :
    Return $m$ .
  Else :
    Return $\perp$ .

---

**Listing 6.8:** Distinguisher algorithm $\mathcal{D}$.

constant factor 2 from the bound.

The algorithm $\mathcal{D}$ simulates Game 2 perfectly for $\mathcal{A}$. However, it also plays the IND-CCA2 game against the **SC_DEM**, and succeeds whenever $\mathcal{A}$ wins. Hence, they have the same advantage. ■

The only event that keeps the IND-CCA2 adversary $\mathcal{D}$ game from becoming the IND-PA adversary used in Dent's original reduction, is precisely the event that causes the INT-CCA2-dependent bound to show up in his reduction: that the adversary queries a *valid* signcryption $(c, e^*)$ to the unsigncryption oracle. This causes a problem because $\mathcal{D}$ does not know the *random* key corresponding to $e^*$, and because responding with $\perp$ to all such queries will cause an error whenever $\sigma$ in fact is a forgery. Dent uses an INT-CCA2 adversary to bound the probability of this event occuring, and therefore gets away with requiring the **SC_DEM** to be IND-PA. However, the event that this occurs is a very specific and restricted case of strong unforgeability. In fact, the implication is that $\mathcal{A}$ has been able to produce a valid encapsulation of a *new* message that is nevertheless *identical* to the challenge encapsulation, and encrypted that message with the associated key. Such an adversary is trivially able to break the INT-CCA2 security of **SC_KEM** (as Dent shows in his proof), but far from all classes of INT-CCA2 forgers are able to complete such a specific forgery. It seems plausible that a weaker notion than INT-CCA2 would suffice to eliminate this event from occuring (and hence be able to use IND-PA in the final reduction step), but the author has not been able to do this.

The main question remaining is whether the tradeoff made by the new reduction is worth its cost. An IND-CCA2 secure signcryption DEM can easily be constructed from an IND-PA secure signcryption DEM together with a secure MAC [CS04]. However, using such a construction needlessly sacrifices some efficiency. First of all, it increases the length of the signcrypted message, and also the computational complexity of the signcryption and unsigncryption algorithms. More subtly, it is conceptually redundant to use a MAC to authenticate the message in the signcryption DEM, since the message is already being authenticated by the associated signcryption KEM (as Theorem 4.2.4 demonstrates). On the positive side, the INT-CCA2 requirement on the signcryption KEM may be weakened to a notion corresponding to the hybrid signcryption scheme being UF-CMA, as the reduction in Theorem 4.2.4 would still be valid when using the weaker security notion on both sides. Furthermore, the overall security reduction for confidentiality will usually be tighter than the original, since the INT-CCA2 dependent term is avoided in the new bound.

A different approach is to modify Dent's framework for hybrid signcryption. An intuitive approach is to replace the notions of IND-CCA2 and INP-CCA2 for SC_KEM with that of indistinguishability of encapsulations, where the adversary is given the $(K^*, e^*) \xleftarrow{R} \mathit{Encap}(x_s, y_r, m_b)$ as input. This does not work, because it is necessary to show that the key output by *Encap* is indistinguishable from a *random* key to be able to bound the advantage in the final game with that of an IND-* adversary against the SC_DEM. Furthermore, this and similar modifications does nothing to escape the problem with the reduction encountered above; that of an adversary asking a valid unsigncryption query containing the challenge encapsulation. Another line of investigation stems from the observation that the requirement of an IND-CCA2 secure signcryption DEM is the same as the situation for IND-CCA2 secure hybrid encryption. Hence, developments in the encryption setting that weaken the security requirement for the DEM, should be analysed to see whether they can be applied to the signcryption setting as

well. A recent paper by Abe et.al. [AGK05] explores the notion of secure hybrid encryption as a combination of a so-called Tag-KEM and a DEM. In this model, the DEM is only required to be IND-PA secure. Whether it is possible to adopt their technique to the signcryption setting is currently a question being investigated.

# 7

# Additional security requirements

In general, specific applications of signcryption may have security requirements above and beyond confidentiality and authenticity. This chapter will briefly mention ways of achieving non-repudiation for Zheng's signcryption scheme, and analyze a modification to Zheng's scheme proposed by Jung et.al. intended to provide forward secrecy [JLLC01].

## 7.1 Non-repudiation of signcryption

As noted in Chapter 3.1, non-repudiation is not automatically guaranteed by the authenticity of signcryption schemes. Although it has not been given the same amount of attention as the security goals of authenticity and confidentiality, Malone-Lee considers general models for signcryption schemes providing non-repudiation in [ML04b] and [ML04a]. His suggested solution is to add a public verification algorithm to the signcryption scheme, that can be used to repudiate a message in a non-interactive fashion. Such an algorithm would only require the message, ciphertext and associated public keys as input. Although Malone-Lee's suggested model is useful to keep in mind when developing signcryption schemes, it is often difficult to construct such an algorithm for many signcryption schemes. For this reason, existing signcryption schemes tend to rely on interactive zero-knowledge protocols to provide non-repudiation. This is also the case for Zheng's scheme.

In Zheng's signcryption scheme, non-repudiation is made difficult by the fact that an untrusted judge is unable to verify that a signcryptext corresponds to a given message directly. There does not appear to be an easy way to construct a non-interactive repudiation algorithm as in Malone-Lee's model. This is because the value $\kappa$ must be computed to be able to relate a signcryptext to its corresponding plaintext, which requires access to either Alice or Bob's private key. In his original paper [Zhe97], Zheng therefore proposes an interactive mechanism where Bob uses a zero-knowledge argument to convince a judge that a message $m$, signcryptext $\sigma = (c, r, s)$ and the associated value $\kappa$ are related to Alice and Bob's public keys in the correct way. Unfortunately, Petersen and Michels demonstrate that a dishonest judge can use this protocol to compromise the confidentiality of future communication between Alice and Bob [PM98]. In the full version of his paper, Zheng gives an alternate zero-knowledge argument based on boolean satisfiability, which appears to provide non-repudiation without damaging the security of his scheme.

## 7.2 Forward security

Forward security is the property that compromise of a secret key does not damage the security of previous communications [AABN02] [MB04]. In the full version of his paper, Zheng

discusses the concept of forward secrecy [Zhe97]. This is an example of forward security, taken with respect to Alice's private key and the confidentiality of signcryptexts. The idea is that an active attacker Mallory is unable to break the confidentiality of previous signcryptexts sent by Alice, even after learning Alice's sending key $x_s$. Zheng concludes that his scheme does not provide forward secrecy: given Alice's private sending key $x_s$, Bob's public receiving key $y_r$, and a signcryptext $\sigma = (c, r, s)$ from Alice to Bob, Mallory may compute the left hand side of $(y_r^{x_s+r})^s \equiv \kappa \equiv (y_s \cdot g^r)^{s \cdot x_r} (\text{mod } p)$ and use $\kappa$ to decrypt and verify the message in the usual fashion.

In 2002, Jung et.al. proposed a simple modification to Zheng's scheme [JLLC01], and claim that it grants forward secrecy with respect to $x_s$. Their idea is to have Alice transmit the ciphertext $\sigma = (c, R, s)$, where $R \leftarrow g^r \mod p$. Their claimed result is that this prevents Mallory from recovering $\kappa$ from the ciphertext, while maintaining the original scheme's properties with regards to unforgeability and indistinguishability. However, the security arguments provided are somewhat ad hoc, and do not constitute a full proof that the modified scheme provides a strong notion of forward secrecy.

## 7.3 A modification of Zheng's signcryption scheme

The modified signcryption scheme $\mathcal{MSC}$ proposed by Jung et. al. [JLLC01] is almost identical to Zheng's signcryption scheme from Definition 3.3.3. The only difference lies in the specifications of the signcryption and unsigncryption algorithms, which are modified in an attempt to provide forward secrecy:

---

$SC(x_s, y_r, m)$:
- $n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
- $\kappa \leftarrow y_r^n \mod p$.
- $bind \leftarrow y_s || y_r$.
- $r \leftarrow \mathcal{H}(m || bind || \kappa)$.
- $s \leftarrow n/(x_s + r) \mod q$.
- $K \leftarrow \mathcal{G}(\kappa)$.
- $c \leftarrow Encr_K(m)$.
- $R \leftarrow g^r \mod p$.
- Return $\sigma \leftarrow (c, R, s)$.

$USC(y_s, x_r, \sigma)$:
- Parse $(c, R, s) \leftarrow \sigma$.
- $\kappa \leftarrow (y_s \cdot R)^{s \cdot x_r} \mod p$.
- $k \leftarrow \mathcal{G}(\kappa)$.
- $m \leftarrow Decr_k(c)$.
- $bind \leftarrow y_s || y_r$.
- $r \leftarrow \mathcal{H}(m || bind || \kappa)$.
- $R' \leftarrow g^r \mod p$.
- If $R' = R$:
  - Return $m$.
- Else:

Return $\perp$.

**Listing 7.1:** The modified signcryption scheme $\mathcal{MSC}$.

The computational cost of the signcryption algorithm in $\mathcal{MSC}$ is increased by one exponentiation as compared to $\mathcal{SC}$, since Alice has to compute $g^r \mod p$. When unsigncrypting, Bob saves an exponentiation when computing $\kappa$, but needs to perform an additional exponentiation during the final verification step. This causes a net increase in computational cost, as it is not longer possible to lower the cost of computing the modular exponentiations (using the technique described in [Zhe97, Appendix B]) when they are not performed simultaneously. The full cost of two modular exponentiations is therefore incurred during unsigncryption. When examining the security of a modified scheme such as $\mathcal{MSC}$, it is essential to ensure that it still secure under the same notions as the original scheme. In the case of authenticity, a direct reduction can be made from each scheme to the other.

**Theorem 7.3.1.** *The signcryption scheme $\mathcal{SC}$ is UF-CMA secure if and only if the modified scheme $\mathcal{MSC}$ is also UF-CMA secure. Concretely, given adversaries $\mathcal{A}$ against the UF-CMA security of $\mathcal{MSC}$ and $\mathcal{B}'$ against the UF-CMA security of $\mathcal{SC}$, one may construct adversaries $\mathcal{B}$ and $\mathcal{A}'$ against $\mathcal{SC}$ and $\mathcal{MSC}$ respectively, such that*

$$\begin{aligned} Adv_{UF-CMA,\mathcal{A}}(k,t,q_{sc}) &\leq Adv_{UF-CMA,\mathcal{B}}(k,O(t)+t_o,q_{sc}) \\ Adv_{UF-CMA,\mathcal{B}'}(k,t',q_{msc}) &\leq Adv_{UF-CMA,\mathcal{A}'}(k,O(t')+t'_o,q_{msc}). \end{aligned} \tag{7.1}$$

*In the above expression, $q_{sc}$ and $q_{msc}$ are the number of signcryption oracle queries asked by $\mathcal{A}$ and $\mathcal{B}'$. The times $t_o$ and $t'_o$ denote the time required by $\mathcal{B}$ and $\mathcal{A}'$ to generate replies to $q_{sc}$ and $q_{msc}$ oracle queries, respectively.*

*Proof.* Consider an adversary $\mathcal{A}$ that breaks the UF-CMA security of $\mathcal{MSC}$. The following algorithm uses $\mathcal{A}$ to break the UF-CMA security of $\mathcal{SC}$:

$\mathcal{B}(I, y_s, x_r; SC\_Oracle)$:
  $bind \leftarrow y_s || y_r$.
  $(m^*, \sigma^*) \xleftarrow{R} \mathcal{A}(I, y_s, x_r; SC\_Sim)$.
  Parse $(c^*, R^*, s^*) \leftarrow \sigma^*$.
  $\kappa^* \leftarrow (y_s \cdot R^*)^{s^* \cdot x_s} \mod p$.
  $r^* \leftarrow \mathcal{H}(m || bind || \kappa^*)$.
  Return $\left(m^*, (c^*, r^*, s^*)\right)$.

$SC\_Sim(m)$:
  $\sigma \leftarrow SC\_Oracle(m)$.
  Parse $(c, r, s) \leftarrow \sigma$.
  Return $(c, g^r \mod p, s)$.

**Listing 7.2:** Construction of UF-CMA adversary against $\mathcal{SC}$.

Given an adversary $\mathcal{B}'$ that breaks the UF-CMA security of $\mathcal{SC}$, an algorithm $\mathcal{A}'$ can be constructed to break the UF-CMA security of $\mathcal{MSC}$.

$\mathcal{A}'(I, y_s, x_r; SC\_Oracle)$ :
$bind \leftarrow y_s || y_r$ .
$(m^*, \sigma^*) \xleftarrow{R} \mathcal{B}'(I, y_s, x_r; SC\_Sim)$ .
Parse $(c^*, r^*, s^*) \leftarrow \sigma^*$ .
Return $\left(m^*, (c^*, g^{r^*} \mod p, s^*)\right)$ .

$SC\_Sim(m)$ :
$\sigma \leftarrow MSC\_Oracle(m)$ .
Parse $(c, R, s) \leftarrow \sigma$ .
$\kappa \leftarrow (y_s \cdot R)^{s \cdot x_s} \mod p$ .
$r \leftarrow \mathcal{H}(m || bind || \kappa)$ .
Return $(c, r, s)$ .

**Listing 7.3:** Construction of UF-CMA adversary against $\mathcal{MSC}$.

It is clear that both $\mathcal{B}$ and $\mathcal{A}'$ simulate the runtime environments of their respective black-box adversaries perfectly, and run in the same time complexity as the adversaries plus the time needed to simulate signcryption oracle queries and perform post-processing steps. Furthermore, the advantages of $\mathcal{A}$ and $\mathcal{B}$ as well as of $\mathcal{A}'$ and $\mathcal{B}'$ are always the same, as $m$ is only asked to the signcryption oracles when it is received as a query by the corresponding signcryption simulator. ■

**Remark 7.3.2.** In the case of sUF-CMA, neither $\mathcal{B}$ nor $\mathcal{A}'$ can be certain that they return a valid forgery given that $\mathcal{A}$ and $\mathcal{B}'$ do. This is because the forgery returned by the black box adversaries are modified. However, if $m$ and $(c, r, s)$ is a valid forgery of $\mathcal{SC}$ that has not been returned by the signcryption oracle simulator, then the probability that $m$ and $(c, R, s)$ is *not* valid forgery of $\mathcal{MSC}$ is at most $\frac{q_{sc}}{q}$, since $\mathcal{H}$ is a random oracle and $g^r \mod p$ is a random permutation on the group. The reverse argument demonstrates that a valid sUF-CMA forgery of $\mathcal{MSC}$ leads to an invalid sUF-CMA forgery of $\mathcal{SC}$ with negligible probability.

It turns out to be more difficult to construct a reduction to show that an IND-CCA2 adversary against $\mathcal{MSC}$ implies the existence of an IND-CCA2 adversary against $\mathcal{SC}$. The reason for this is that the IND-CCA2 adversary against $\mathcal{SC}$ is unable to process unsigncryption oracle requests on the form $(c, g^r \mod p, s)$, since it can neither compute $r$ directly nor compute $\kappa$ and use $\mathcal{H}$ to obtain $r$. On the other hand, it is quite obvious that when there does not exist an efficient algorithm to distinguish between the triplets $(c_0, r_0, s_0) \xleftarrow{R} SC(x_s, y_r, m_0)$ and $(c_1, r_1, s_1) \xleftarrow{R} SC(x_s, y_r, m_1)$[1], then it is hard to distinguish $(c_0, g^{r_0} \mod p, s_0)$ from $(c_1, g^{r_1} \mod p, s)$ as well . This intuition may indeed be confirmed by performing a game hopping proof to bound the probability that $\kappa^*$ is not asked to a random oracle. The proof can be performed similarly to that of Theorem 5.2.8, with the obvious modifications[2].

Having established that the modified signcryption scheme is not significantly more *insecure* than the original with respect to authenticity and confidentiality, it is necessary to examine the claim that it grants forward security. According to Jung et.al. [JLLC01], it is not feasible to

[1] I.e. that $\mathcal{SC}$ satisfies a notion of indistinguishability.

[2] Alternately, one may prove that **MSC_KEM**, defined in the obvious manner, is IND-CCA2 and INP-CCA2 secure. The method of proof is still the same.

compute the value $\kappa$ from an $\mathcal{MSC}$ signcryptext $\sigma$ from Alice to Bob $\sigma$, given Alice's private sending key $x_s$ and Bob's public receiving key $y_r$.

**Proposition 7.3.3.** *Suppose that there exists an algorithm $\mathcal{A}$ who can compute the value $\kappa \equiv (y_s \cdot R)^{s \cdot x_r} \equiv (y_r^{x_s+r})^s (mod\ p)$ in the Schnorr group specified by parameters $(p, q, g)$, when given the values $x_s$, $y_r$, $R$ and $s$ as input. Then there exists an efficient algorithm $\mathcal{B}$ to solve the Computational Diffie-Hellman problem in that group.*

*Proof.* Consider the following adversary $\mathcal{B}$ against the CDH problem:

---

Public information:
  Security parameters $k$, $k_q$.
  $k$–bit prime $p$, $k_q$–bit prime $q$ such that $q|(p-1)$.
  Element $g \in \mathbb{Z}_p$ of order $q$.
  Public information for $\mathcal{MSC}$, $I$.

$\mathcal{B}(X, Y)$:
  $(x_s, y_s) \xleftarrow{R} Key_s(I)$.
  $s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
  $R \leftarrow X$.
  $y_r \leftarrow Y$.
  $\kappa \xleftarrow{R} \mathcal{A}(x_s, y_r, R, s)$.
  $Z \leftarrow y_r^{-x_s} \cdot \kappa^{\frac{1}{s}} \mod p$.
  Return $Z$.

---

**Listing 7.4:** Forward security of $\mathcal{MSC}$.

The correct value $\kappa$ that is to be returned by $\mathcal{A}$ is by construction equal to $(y_r^{x_s+r})^s \mod p$. This is equal to $(y_r^{x_s} \cdot y_r^r)^s \mod p$. From the way $\mathcal{B}$ is constructed, the value $y_r^r$ is the solution to the CDH problem instance $(X, Y)$, while $y_r$, $x_s$ and $s$ are known. Hence, $\mathcal{B}$ can compute the correct CDH solution $Z$ and thereby win its game whenever $\mathcal{A}$ succeeds. The runtime environment of $\mathcal{A}$ is simulated perfectly, since all four input variables provided by $\mathcal{B}$ have same distributions as that of an arbitrary signcryptext. ∎

A further argument that $\mathcal{MSC}$ provides full forward secrecy, relies on the tacit assumptions that the symmetric encryption scheme used is IND-PA, and that all hash functions are modelled as random oracles. Any breach of confidentiality for a single signcryptext must therefore require $\mathcal{A}$ to recover $\kappa$. However, the above analysis does not take into account the possibility that having access to several valid signcryptions sent from Alice may enable an adversary to break the confidentiality of the scheme, nor the possibility of a partial information leakage. It is not clear whether $\mathcal{A}$ by examining several relations of the type $(y_s \cdot R_i)^{s_i \cdot x_r} \equiv (y_r^{x_s+r_i})^{s_i} (\mathrm{mod}\ p)$ might be able to recover either $x_r$ or at least one of the $r_i$.

Ideally, $\mathcal{MSC}$ should be shown to achieve the notion of indistinguishability of signcryptions, against an adversary who knows Alice's private sending key $x_s$ and has access to a (possibly flexible) unsigncryption oracle. This is not considered in [JLLC01] at all. Although this may seem to be a reasonable conjecture based on the previous result and what has been shown

about the security of $\mathcal{SC}$ in Chapters 5 and 6, a proof does not exist. Such a proof appears difficult to construct due to the limited freedom on part of a simulator, as the only parameter not known by $\mathcal{A}$ is Bob's private receiving key $x_r$. Therefore, the only conclusion that can be made is that $\mathcal{MSC}$ grants *some* degree of provable forward secrecy. Whether this is worth the computational cost incurred relative to $\mathcal{SC}$ is dubious, but will depend on the intended application of the scheme.

# 8

# Concluding remarks

In this final chapter, a brief summary of the results obtained in the thesis will be given, together with a discussion about possible directions for further research.

## 8.1 Summary of results

In its examination of signcryption schemes and security models for signcryption, the focal point of this thesis has been Zheng's original signcryption scheme. This has served the dual purpose of illuminating the assumptions on which the security of the scheme rests, while simultaneously highlighting some of the differences between Dent's new framework for hybrid signcryption and older models. The previous proof of security for Zheng's scheme by Baek, Steinfeld and Zheng [BSZ02] has been given a detailed re-examination in Chapter 5, making their results both stronger and more accessible. Meanwhile, Dent's claim that Zheng's scheme is an insider secure hybrid encryption scheme has been verified [Den04] in Chapters 6.2 and 6.3. Having performed security proofs for Zheng's signcryption scheme under two different security models opens for a direct comparison of the results obtained. The relative strengths of the reductions show that Dent's hybrid model provides strong security guarantees with respect to authenticity of signcryption, but that a better reduction for the confidentiality of hybrid signcryption may be desirable. An alternate confidentiality reduction for hybrid signcryption has been attempted in Chapter 6.5, but it is unclear whether the result obtained provides a benefit over Dent's original reduction. Finally, a suggested modification of Zheng's scheme alleged to provide forward security with respect to confidentiality of signcryption and Alice's private key [JLLC01] has been given a brief examination in Chapter 7. The claim of forward forward has been demonstrated to be correct in a limited sense, but a full-fledged security proof has not yet been established.

## 8.2 Topics for further research

Dent's hybrid approach gives a clean and structured approach to analysis of different aspects of a secure signcryption scheme. However, the advantage bound for the confidentiality of hybrid signcryption reached in the model appears to be less tight than one might hope for. Further development of the hybrid model to accommodate tighter generic security reductions for hybrid signcryption is therefore desirable. One concrete line of investigation is whether the Tag-KEM/DEM model for hybrid encryption, recently developed by Abe et.al. [AGK05], may be adapted to fit hybrid signcryption schemes.

Another question that warrants further investigation is the relationship between strong and regular existential unforgeability. As demonstrated in Chapter 6.1, the SDSS1 signature

scheme upon which Zheng's signcryption scheme is constructed satisfies the stronger notion of existential unforgeability in the Random Oracle Model. However, there appears to be little knowledge about how the two notions relate for common signature schemes. This particularly applies to signature schemes constructed by applying the Fiat-Shamir heuristic to an identification scheme, in the Random Oracle Model.

Finally, it is interesting to note that the only well-known example of a hybrid signcryption scheme is Zheng's original proposal, even 8 years after the signcryption primitive was introduced. Although Zheng's scheme is efficient and provably secure, it would be interesting to examine the properties of alternate constructions. As a first step, it might be useful to consider hybrid signcryption based on a signature scheme whose unforgeability can be given a tight reduction to its underlying problem, such as [CM05]. However, an even more intriguing possibility is whether one can construct a signcryption scheme that is provably secure outside the Random Oracle Model. Other interesting possibilities include the construction and analysis of insider secure hybrid signcryption schemes in the identity based setting, or based on a different underlying problem than discrete logarithms.

# Bibliography


[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332, pages 418–433. Springer–Verlag, 2002.

[ADR02] 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.

[AGK05] Masayuki Abe, Rosario Gennaro, and Kaoru Kurosawa. Tag-KEM/DEM: A new framework for hybrid encryption. Cryptology ePrint Archive, Report 2005/027, 2005. http://eprint.iacr.org/2005/027/.

[An01] Jee Hea An. Authenticated encryption in the public-key setting: Security notions and analyses. Cryptology ePrint Archive, Report 2001/079, 2001. http://eprint.iacr.org/2001/079/.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027, pages 171–188. Springer–Verlag, 2004.

[BDJR97] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of FOCS '97*, pages 394–424. IEEE, 1997.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – CRYPTO '98*, volume 1462, pages 26–45. Springer–Verlag, 1998.

[Bel99] Mihir Bellare. Practice-oriented provable security. In *Lectures on Data Security: Modern Cryptology in Theory and Practice*, volume 1561, pages 1–15. Springer–Verlag, 1999.

[Ber05] Daniel J. Bernstein. Cache-timing attacks on AES, 2005. Available at http://cr.yp.to/papers.html#cachetiming.

[BFMLS05] Kamel Bentahar, Pooya Farshim, John Malone-Lee, and Nigel Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058, 2005. http://eprint.iacr.org/.

[BGB04] Nikita Borisov, Ian Goldberg, and Eric Brewer. Off-the-record communication, or, why not to use PGP. In *Proceedings of WPES '04*, pages 77–84. ACM Press, 2004.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR04] Mihir Bellare and Phillip Rogaway. The game-playing technique. Cryptology ePrint Archive, Report 2004/331, 2004. http://eprint.iacr.org/2004/331/.

[BSZ02] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *Proceedings of PKC 2002*, volume 2274, pages 80–98. Springer–Verlag, 2002.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. Cryptology ePrint Archive, Report 1998/011, 1998. http://eprint.iacr.org/1998/011/.

[CM05] Benoit Chevallier-Mames. An efficient cdh-based signature scheme with a tight security reduction. Cryptology ePrint Archive, Report 2005/035, 2005. http://eprint.iacr.org/2005/035/.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology – CRYPTO '98*, volume 1462, pages 13–25. Springer–Verlag, 1998.

[CS04] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[Den02] Alexander W. Dent. A designer's guide to KEMs. Cryptology ePrint Archive, Report 2002/174, 2002. http://eprint.iacr.org/2002/174/.

[Den04] Alexander W. Dent. Hybrid cryptography. Cryptology ePrint Archive, Report 2004/210, 2004. http://eprint.iacr.org/2004/210/.

[Den05] Alexander W. Dent. Hybrid signcryption schemes with insider security, 2005. To appear at ISC 2005.

[DFJW04] Yevgeniy Dodis, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish. Optimal signcryption from any trapdoor permutation. Cryptology ePrint Archive, Report 2004/020, 2004. http://eprint.iacr.org/2004/020/.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263, pages 186–194. Springer–Verlag, 1987.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker hiding all partial information. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, pages 365–377. ACM, May 1982. http://theory.lcs.mit.edu/ cis/pubs/shafi/1982-stoc.pdf.

[GMR84] Shafi Goldwasser, Silvia Micali, and Ronald Rivest. A paradoxical solution to the signature problem. In *Proceedings of FOCS 1984*, pages 441–448. IEEE, 1984.

[Gol01] Oded Goldreich. *The Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[JLLC01] Hee Yun Jung, Dong Hoon Lee, Jong In Lim, and Ki Sik Chang. Signcryption schemes with forward secrecy. In *Proceedings of WISA '01*, volume 2, pages 403–415, 2001.

[Kah96] David Kahn. *The Codebreakers*. Scribner, 1996. Revised and updated edition.

[KM04] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". Cryptology ePrint Archive, Report 2004/152, 2004. http://eprint.iacr.org/2004/152/.

[LQ04] Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from Gap Diffie-Hellman groups. In *Proceedings of PKC 2004*, volume 2947, pages 187–200. Springer–Verlag, 2004.

[MB04] Noel McCullagh and Paulo S. L. M. Barreto. Efficient and forward-secure identity-based signcryption. Cryptology ePrint Archive, Report 2004/117, 2004. http://eprint.iacr.org/2004/117/.

[ML04a] John Malone-Lee. *On the Security of Signature Schemes and Signcryption Schemes*. PhD thesis, University of Bristol, 2004. http://www.cs.bris.ac.uk/Publications/Papers/2000104.pdf.

[ML04b] John Malone-Lee. Signcryption with non-interactive non-repudiation. Technical Report CSTR-02-004, Department of Computer Science, University of Bristol, 2004. http://www.cs.bris.ac.uk/Publications/Papers/1000628.pdf.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1996.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In *Advances in Cryptology - CRYPTO '98*, volume 1462, pages 354–369. Springer–Verlag, 1998.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *Progress in Cryptology - CT-RSA 2001*, volume 2020, pages 104–118. Springer–Verlag, 2001.

[PM98] Holger Petersen and Markus Michels. Cryptanalysis and improvement of signcryption schemes. volume 145 (2), pages 149–151. IEE, 1998.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 387–398. Springer–Verlag, 1996.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[Rab79] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979. http://www.lcs.mit.edu/publications/pubs/pdf/MIT-LCS-TR-212.pdf.

[Sch96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., second edition, 1996.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:656–715, 1949. http://www.cs.ucla.edu/ jkong/research/security/shannon.html.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/2001/112/.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332/.

[SPMLS02] Jacques Stern, David Pointcheval, John Malone-Lee, and Nigel P. Smart. Flaws in applying proof methodologies to signature schemes. In *Advances in Cryptology - CRYPTO 2002*, volume 2442, pages 93–110. Springer–Verlag, 2002.

[Sta94] National Bureau of Standards. Digital Signature Standard. Technical Report FIPS Publication 186, National Bureau of Standards, 1994.

[WDKM04] Guilin Wang, Robert H. Deng, Dong Jin Kwak, and Sang Jae Moon. Security analysis of two signcryption schemes. In *Proceedings of ISC 2004*, volume 3225, pages 123–133. Springer–Verlag, 2004.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. To appear at CRYPTO '05, 2005. Preliminary abstract available at http://theory.csail.mit.edu/ yiqun/shanote.pdf.

[Zhe97] Yuliang Zheng. Digital signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption). In *Advances in Cryptology – 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.