Design and Analysis of Opaque Signatures

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Laila El Aimani

aus

Marrakech, Marokko

Bonn 2011

Tag der Disputation: 29 April 2011
Promotionskommission:

Prof. Dr. Joachim von zur Gathen, Erstgutachter - Betreuer (b-it, Universität Bonn)

Prof. Dr. Marek Karpinski, Fachnahes Mitglied (Universität Bonn)

Prof. Dr. Alexander Markowetz, Fachangrenzendes Mitglied (Universität Bonn)

Prof. Dr. Kenny Paterson, Zweitgutachter (Royal Holloway, University of London)

Prof. Dr. Jean-Jacques Quisquater, Fachangrenzendes Mitglied (Université Catholique de Louvain)

Erscheinungsjahr: 2011

Abstract (Zusammenfassung)

Digital signatures were introduced to guarantee the authenticity and integrity of the underlying messages. A digital signature scheme comprises the key generation, the signature, and the verification algorithms. The key generation algorithm creates the signing and the verifying keys, called also the signer's private and public keys respectively. The signature algorithm, which is run by the signer, produces a signature on the input message. Finally, the verification algorithm, run by anyone who knows the signer's public key, checks whether a purported signature on some message is valid or not. The last property, namely the universal verification of digital signatures is undesirable in situations where the signed data is commercially or personally sensitive. Therefore, mechanisms which share most properties with digital signatures except for the universal verification were invented to respond to the aforementioned need; we call such mechanisms "opaque signatures". In this thesis, we study the signatures where the verification cannot be achieved without the cooperation of a specific entity, namely the signer in case of *undeniable signatures*, or the confirmer in case of *confirmer signatures*; we make three main contributions.

We first study the relationship between two security properties important for public key encryption, namely data privacy and key privacy. Our study is motivated by the fact that opaque signatures involve always an encryption layer that ensures their opacity. The properties required for this encryption vary according to whether we want to protect the identity (i.e. the key) of the signer or hide the validity of the signature. Therefore, it would be convenient to use existing work about the encryption scheme in order to derive one notion from the other.

Next, we delve into the generic constructions of confirmer signatures from basic cryptographic primitives, e.g. digital signatures, encryption, or commitment schemes. In fact, generic constructions give easy-to-understand and easy-to-prove schemes, however, this convenience is often achieved at the expense of efficiency. In this contribution, which constitutes the core of this thesis, we first analyze the already existing constructions; our study concludes that the popular generic constructions of confirmer signatures necessitate strong security assumptions on the building blocks, which impacts negatively the efficiency of the resulting signatures. Next, we show that a small change in these constructions makes these assumptions drop drastically, allowing as a result constructions with instantiations that compete with the dedicated realizations of these signatures.

Finally, we revisit two early undeniable signatures which were proposed with a conjectural security. We disprove the claimed security of the first scheme, and we provide a fix to it in order to achieve strong security properties. Next, we upgrade the second scheme so that it supports a

desirable feature, and we provide a formal security treatment of the new scheme: we prove that it is secure assuming new reasonable assumptions on the underlying constituents.

Acknowledgments

In the course of my PhD studies, I have had the support of many people and I am pleased to acknowledge here their different contributions.

My largest debt goes undoubtedly to my supervisor Joachim von zur Gathen for giving me the opportunity to pursue this PhD in a splendid environment. I thank him also for his insights, guidance, and support at all stages of my PhD studies.

Next, I am profoundly grateful to Damien Vergnaud for making me discover cryptographic protocols in general and undeniable signatures in particular. His advices and suggestions have contributed inestimably to my understanding of this fascinating area.

I have also benefited from conversations and correspondence with many researchers that I met at different events. I wish to thank particularly Marc Joye, Pascal Paillier, and Kenny Paterson for precious discussions which sharpened my knowledge in reductionist security.

I would like to thank also Prof. Marek Karpinski, Prof. Alexander Markowetz, Prof. Kenny Paterson, and Prof. Jean-Jacques Quisquater for accepting to devote a good part of their time in order to evaluate this work. I actually appreciate greatly the extremely careful assessment of my thesis and the many interesting remarks I got from my reviewers. It is also an honor for me to have such brilliant people in my PhD committee.

Accomplishing this PhD would not have been possible without the assistance of my colleagues at cosec and b-it. I cherish so much our seminars, discussions, and social activities which made my PhD experience both interesting and enjoyable.

My sincere thanks go also to my current colleagues at Technicolor, more specifically the Security and Content Protection Labs team at Rennes. The head of this group, Eric Diehl, deserves special thanks for his interest in my work, and for maintaining a rich and stimulating working environment in a friendly atmosphere. I wish to present again my profound gratitude to my colleague Marc Joye; I value very much his technical insight, his availability to discuss my results, and his encouragement to further improve them.

Finally, and at every stage of my studies, I have had endless support from my family. In particular, my mother as well as my sister Salma and my brother Jabrane were an unfailing source of encouragement at times when I was most discouraged. I am heartily thankful for their love and support, and it is to them that I dedicate this work.

Preface

This thesis presents the ensemble of my PhD results obtained in the area of "opaque" signatures, namely [El Aimani & Vergnaud, 2007; El Aimani, 2008, 2009a,b, 2010][1].

A digital signature is a mechanism that captures most properties satisfied by a "traditional" signature in the paper world. In fact, digital signatures guarantee that the signed message has not been altered in transit, and that it comes from the source that claims to be its provenance, namely the signer. More formally, a digital signature consists of three algorithms: (1) the key generation algorithm which creates the signing and the verifying keys, (2) the signing algorithm which takes on input the signing key and a message, and outputs a signature on the input message, (3) and the verification algorithm which checks the validity of an alleged signature on a given message using the verifying key. An important feature in digital signatures is the universal verification, i.e. anyone who knows the verifying key, called also the signer's public key, can verify signatures issued by this signer. However, such a property can be undesirable in some applications and needs to be controlled or limited; we talk then about obscure or *opaque* signatures. In this document, we will focus on confirmer and undeniable signatures. Let us then specify the context.

Consider for example the case of inter-organizational electronic messages; signatures on these messages are indispensable to resolve disputes as they ensure integrity and authenticity of the underlying messages, however, self-authentication of these signatures will make the messages vulnerable to industrial spy or extortionist. Undeniable signatures come to rescue in this situation as they: (1) cannot be verified without cooperation with the signer via the confirmation/denial protocols, (2) are non-transferable since a verifier cannot transfer his conviction, to a third party, about the validity/invalidity of a signature he has just verified, (3) are binding in the sense that a signer cannot deny a signature he has actually issued. Unfortunately, the very virtue of undeniable signatures (verification with only the signer's help) became their major shortcoming for many practical applications since absence of the signer obstructs the entire verification process. Therefore, the concept of undeniable signatures was upgraded to designated confirmer signatures where the verification is *delegated* to a designated confirmer.

Building complicated systems upon simple and basic primitives is customary in cryptography as it allows to re-use existing work about the primitives, and it achieves easy-to-understand and

[1] The works [El Aimani & von zur Gathen, 2007; El Aimani & Raekow, 2009, 2010] are not reported in this document as they do not accord with the general theme of the thesis.

easy-to-prove systems. However, monolithic or dedicated schemes better often, in terms of efficiency, those obtained from instantiating generic constructions with concrete primitives. This is mainly due to the fact that generic constructions cannot in general use the specific properties of their underlying components in order to optimize the resulting structure. A tantalizing challenge will be the design of generic constructions of undeniable/confirmer signatures which find practical instantiations with the popular cryptographic primitives. This is the main purpose of this thesis.

Contributions and organization of the document

Apart from the first chapter on the cryptographic tools that will be used throughout the document, we can group the contributions of this thesis in the following classes:

1. **Public key encryption**. In this contribution, detailed in Chapter 2, we study the relationship between two security notions important for public key encryption, namely *data privacy* or *indistinguishability*, which refers to the hardness of distinguishing ciphertexts based on the underlying *data*, and *key privacy* or *anonymity*, which denotes the hardness of distinguishing ciphertexts based on the underlying *(public) key*. We also define the anonymity notion for two popular structures used to build public key encryption schemes, that are key and data encapsulation mechanisms, and we study similarly the connection between this new notion and indistinguishability in these mechanisms. Our work was inspired from a similar work on undeniable signatures, and it is motivated by the fact that opaque signatures involve always an encryption layer to ensure their opacity. The properties that this encryption layer should meet vary according to whether we want to hide the identity of the signer or the validity of the signature. Hence, the need for such a study which specifies easy-to-check properties on any encryption so that data privacy yields key privacy and vice versa, allowing consequently to use existing results about the system instead of doing the work from scratch.

2. **Generic constructions of confirmer signatures**. This contribution constitutes the core of this thesis, and it is described in Part II. More precisely:

   - In Chapter 3, we define the model (syntax of confirmer signatures and security properties) we adhere to in our work. Moreover, we survey the different generic constructions of confirmer signatures found in the literature. Most such proposals follow either the sign-then-encrypt or the commit-then-sign paradigms. We also re-write some of the security proofs of these constructions so that they stay resilient against some malicious adversaries, and we provide other proofs which were due to appear in forthcoming papers of the corresponding authors but were not given so far.
   - In Chapter 4, we analyze and improve the confirmer signatures obtained from the sign-then-encrypt technique. In a nutshell, this method consists in first producing a digital signature on the message to be signed, then encrypting the resulting signature. This

method originally required the constituents, that are the underlying signature and encryption schemes, to meet the highest notions of security. In this chapter, we show that the requirement on the signature scheme is also necessary for the security of the construction, whilst the condition on the encryption could be weakened. However, we prove also the necessity of this weakened condition, which translates in excluding a useful type of encryption. Next, we circumvent this problem by modifying slightly the paradigm so that it accepts a cheap and useful type of encryption, namely homomorphic encryption. We demonstrate the efficiency of the resulting construction by explicitly describing the confirmation and denial protocols, a task which has not been addressed in all generic constructions of confirmer signatures which implement the sign-then-encrypt principle.

- In Chapter 5, we analyze the second popular method used to devise confirmer signatures, namely the commit-then-sign paradigm which consists in first producing a commitment on the message to be signed, encrypting the string used for the commitment, and finally signing the latter. Similarly to the previous chapter, we show that the paradigm, when used in its basic form, necessitates a strong encryption which renders the construction inefficient or accept very limited instantiations. However, a small change of the basic paradigm makes the assumption on the encryption drop drastically, allowing as a result many practical instantiations. Finally, we shed light on a subcase of the paradigm, that is the encrypt-then-sign paradigm. Such a method provides very efficient confirmer signatures provided there exist efficient non-interactive variants of the underlying confirmation protocol; this is not a problem nowadays due to the progress made recently in this area.

3. **Undeniable signatures**. This part comprises three chapters namely:

   - Chapter 6, where we browse through the different realizations of undeniable signatures. In fact, while the literature on confirmer signatures was more focused on how to obtain them from basic cryptographic primitives, the literature on undeniable signatures was very diverse. We chose to give this survey in order to better situate our work on undeniable signatures that comes in the following two chapters.
   - Chapter 7, where we revisit the undeniable signatures of Damgård and Pedersen. These signatures were proposed in 1996 with a conjectured security that was reported a decade later in a construction of undeniable signatures following the same spirit. In this chapter, we disprove the conjectured security of Damgård and Pedersen's undeniable signatures, and we propose a repair to the scheme which turns out to be an instantiation of the construction proposed earlier in Chapter 4.
   - Chapter 8, where we revisit the undeniable signatures of Michels, Petersen, and Horster that were proposed in 1996, and had also a speculative security. We first modify slightly these signatures so that they support an additional feature, called *gradual conversion*,

which informally means the possibility of converting a set of undeniable signatures pertained to a given event to publicly verifiable ones. Next, we formally prove that the security of our recast rests on new reasonable assumptions that we introduce for the underlying hash function family.

Before ending this preamble, we wish to alert the reader to the importance of carefully checking the security model in which the systems presented in this document are proclaimed to be secure/insecure. In fact, security models can differ very slightly in their definitions, but the repercussions of these smallish differences can be huge; a system secure in one model can be be totally broken in another. Also, a security property which is impossible to reach for a scheme in a model can be easily met in another. This actually reflects one of the main challenges in nowadays cryptography: proposing efficient schemes which achieve strong security properties based on the hardness of well-studied problems .

Notations

General

$\lvert a \rvert$	absolute value of the real number a
$\max_A \mathrm{Exp}$	maximum of the expression Exp when the variable A ranges all the possible values
$\min_A \mathrm{Exp}$	minimum of the expression Exp when the variable A ranges all the possible values
$\log x$	logarithm of x with respect to some unspecified base
$\ln n$	logarithm of x in the base $e = \sum_{n=0}^{\infty} 1/n!$
$x \leftarrow y$ (for variables[2] x, y)	assigning the value of y to x
$[a, b]$	closed interval, i.e. the set of real numbers x in the range $a \leq x \leq b$
(a, b)	open interval, i.e. the set of real numbers x in the range $a < x < b$
$[\![a, b]\!]$	the set of integers x in the range $a \leq x \leq b$
$\lfloor x \rfloor$	floor of the real number x
$a << b$	a is strictly of smaller order than b
$a >> b$	a is strictly of larger order than b

Bit strings

ϵ	empty string
$\overline{a}$	bit complement of the string a
$a \| b$	concatenation of the strings a and b
$\{0,1\}^n$	set of n-bit strings
$\{0,1\}^*$	set of all finite binary strings

Sets

$\emptyset$	empty set
$\#A$	cardinality of the set A
$a \in A$	a is an element of the set A
$a \notin A$	a is not an element of the set A

[2] the symbol $\leftarrow$ has different interpretations according to the context

$A \subset B$	set A is contained in set B
$A \subseteq B$	set A is contained in or equal to set B
$A \cup B$	union of sets A and B
$A \cap B$	intersection of sets A and B
$A \backslash B$	difference of sets A and B
$A \times B$	Cartesian product of sets A and B
$\mathbb{N}$	set of natural numbers
$\mathbb{Z}$	set of integers
$\mathbb{Q}$	set of rational numbers
$\mathbb{R}$	set of real numbers
$\mathbb{Z}_N$	set of integers modulo N (denoted also the set $\mathbb{Z}/N\mathbb{Z}$)
$\mathbb{Z}_N^\times$	group of units in $\mathbb{Z}_N$
$\mathbb{F}_q$	finite field of cardinality q
$\overline{\mathbb{F}_q}$	algebraic closure of $\mathbb{F}_q$
$\mathbb{F}_q^\times$	multiplicative group of $\mathbb{F}_q$

Groups

$(\mathbb{G}, +)$	group $\mathbb{G}$ is denoted additively
$(\mathbb{G}, \cdot)$	group $\mathbb{G}$ is denoted multiplicatively
$0_\mathbb{G}$	identity in $(\mathbb{G}, +)$
$1_\mathbb{G}$	identity in $(\mathbb{G}, \cdot)$
a^{-1}	inverse of element a in a group denoted multiplicatively
$\langle g \rangle$	group generated by the element g
$\mathsf{DL}_g(y)$	discrete logarithm of the group element y in the base g

Functions

$f \colon A \to B$	f is function from set A to set B
$a \mapsto b$	a is mapped to b (by some function)
f^{-1}	inverse of bijective function f
poly	polynomial function
negl	negligible function, i.e. a function of order smaller than the inverse of any polynomial function

Integers

$a \text{ rem } b$	remainder of the Euclidian division of a by b ($b \neq 0$)
$a \vert b$	a divides b
$\gcd(a, b)$	greatest common divisor of integers a and b
$a = b \bmod n$	a is congruent to b modulo n

$a^{-1} \bmod n$	multiplicative inverse of a modulo n
$\Phi(n)$	Euler's totient function

Events, probabilities, and statistics

$\neg E$	complement of event E
$E_1 \wedge E_2$	intersection of event E_1 and event E_2
$E_1 \vee E_2$	union of event E_1 and event E_2
$\Pr[E]$	probability of event E
$\Pr[E_1 \mid E_2]$	probability of event E_1 given event E_2
$a \leftarrow D$ (for a distribution D)	a is sampled from distribution D
$a \xleftarrow{R} S$ (for a finite set S)	(denoted also $a \in_R S$) a is selected uniformly at random from set S

Acronyms

ANO	anonymity
CCA	chosen ciphertext attack
CDCS	convertible designated confirmer signature
CDH	computational Diffie-Hellman
CMA	chosen message attack
CPA	chosen plaintext attack
DDH	decisional Diffie-Hellman
DEM	data encapsulation mechanism
EUF	existential unforgeability
FDH	full domain hash
GDH	gap Diffie-Hellman
HVZK	honest verifier zero-knowledge
IND	indistinguishability
INV	invisibility
KEM	key encapsulation mechanism
NIZK	non-interactive zero-knowledge
NM	non-malleability
OW	one wayness
PCA	plaintext checking attack
PoK	proof of knowledge
PPTM	probabilistic polynomial Turing machine
ROM	random oracle model
SEUF	strong existential unforgeability
SINV	strong invisibility
SRSA	Strong RSA
WHPOK	witness hiding proof of knowledge

ZK	zero-knowledge
ZKIP	zero-knowledge interactive proof

Computability

$\mathcal{A}^{\mathfrak{O}}$	$\mathcal{A}$ has access to the oracle $\mathfrak{O}$
$\mathfrak{O} : a \longmapsto b$	the oracle $\mathfrak{O}$ gets a as a query and responds with b
$a \leftarrow \mathcal{A}(x)$	$\mathcal{A}$ outputs the value a on input x
$\mathcal{I}$	state information
$(P, V)(x)$	two-party protocol (pair of interactive Turing machines) with common input x
$P \xrightarrow{c} V$	P sends c to V
$P \xleftarrow{c} V$	P gets c from V
$P \xleftrightarrow{\mathsf{PoK}} V$	P and V run the interactive proof PoK

Contents

Part I

Preliminaries

Chapter 1

The Theory of Cryptography

Abstract. Cryptology evolved from a crossing where branches of mathematics, computer science, and electrical engineering deposit their contributions, to an autonomic and mature science. In fact, cryptology inherited techniques from various sciences, successfully reshaped and merged them with new concepts to result in a self-contained science, capable of constructing and analyzing systems meeting the imperishable trilogy of requirements: confidentiality, authenticity, and integrity.
In this chapter, we recall aspects of the theory of cryptography that are necessary for this thesis. We start by reminding some important results from complexity theory, a branch of theoretical computer science where cryptography has scooped up many concepts. Next, we recite the basic primitives upon which are based more sophisticated cryptographic systems. Then, we proceed to the description of three of the theoretical pillars that found modern cryptography, namely reductionist security, zero knowledge, and bilinear maps.

1.1 Reminders in complexity theory

Complexity theory is a branch of computer science concerned with the study of fundamental principles of computation. It is a vibrant area of research due to its ubiquity in many different fields: biologists studying models for neuron nets or evolution, electrical engineers developing switching theory to improve hardware design, mathematicians working on foundations of logics and arithmetics, linguists investigating grammars for natural languages, physicists studying the feasibility of building quantum computers, and of course computer scientists seeking efficient algorithms to solve important problems.

In this section, we recall some basics of complexity theory. We refer to the book [Papadimitriou, 1994] for a comprehensive study of this theory.

1.1.1 Symbols, alphabets, languages and problems

A *symbol* is an atomic entity. Examples of frequently used symbols are letters or digits. A *string* is a finite sequence of juxtaposed symbols. The length of a string s is often denoted $|s|$, and consists of the number of symbols composing the string. One special string is the string consisting of zero symbols; it is denoted ϵ and is called the empty string. A string is said to be the concatenation of two strings s and t if it is formed by writing s followed by t.

An *alphabet* is a finite set of symbols. A *language* is a set of strings of symbols from some alphabet. For instance, if Σ is a given alphabet, $\Sigma^\star$ denotes the language consisting of all possible strings composed of symbols in Σ. One alphabet that will occur often in this document is the alphabet $\{0, 1\}$.

A *problem* is intrinsically associated to a certain question; for example computing the greatest common divisor of two integers. Once one specifies values for the input, one obtains an *instance* of the problem to which corresponds some values forming the solutions to the input. Therefore, if we formulate the possible inputs and outputs of a given problem as strings over some alphabet Σ, a problem can be viewed as a subset of $\Sigma^\star \times \Sigma^\star$. In fact, we assume that for every input question $q \in \Sigma^\star$, there exists an output answer $r \in \Sigma^\star$, for instance we consider "no solution" also a possible answer. An important category of problems is that consisting of problems that accept only two possible answers {"yes", "no"}, i.e. the so-called *decision problems*. We can simplify the representation of decision problems by considering only the "language" consisting of questions that have "yes"-answers. We say that a system, e.g. computer, decides a decision problem if it identifies successfully the positive instances, i.e. the questions having "yes"-answers. Finally, decision problems arise very often in complexity theory, and one is especially interested in knowing whether a given decision problem can be decided by some computer or not. To answer such a question, one needs to introduce a formal and universal model of computer.

1.1.2 Computability & Turing machines

A *computation* is informally speaking a process by which one obtains an answer to a certain question. A computation requires a system which performs the computation. This system or "computer" will move from an initial state, which is independent of the question, to a final state where it outputs the answer, if any. A fundamental problem has been the universalization of the computation model. i.e. provide a model for every "computable" function that computes an answer (if it exists) to any question from the set of possible questions. The non-trivial part of this task lies in having to define a model of a "computer", restricted by known physical laws, to perform any kind of computations. Nevertheless, all computational models that have been developed so far, were shown to be equivalent to a very simple model, the so-called Turing machine. This lead Church and Turing to conjecture in 1936 that every computable function can be computed by a Turing machine.

The basic Turing machine has an input tape comprised of infinitely many cells, and a tape head

which scans one cell of the tape at a time. Each cell contains a symbol from a finite alphabet intrinsic to the machine. Initially, the tape contains only the input of the problem (the question), the rest of the tape being blank. Throughout the computations, the configuration of the machine ranges a finite set of states. Finally, if the machine halts (reaches one of the final states), it writes the output on its tape. The machine operates sequentially: as long as it does not reach the final state, it performs one operation, at a time, which depends solely on the current state and the symbol pointed to by its head. An operation of the machine can be either a change of the current state, a writing/overwriting of a symbol on the scanned cell or a move (in both directions) of the head. The function that maps a pair consisting of a symbol and a state to an operation is called the program of the machine. Finally, a language accepted or decided by a Turing machine TM is the set of strings, composed from symbols of the machine alphabet, which cause the TM to enter a final state.

1.1.3 Extended Turing machines

Multi-tape Turing machines

A multi-tape Turing machine is comprised of k tape heads and k tapes, each has an infinite number of cells. A configuration of the machine at some time point consists of the current state and of the positions of the k tape heads. Similarly to single-tape Turing machines, one operation of a multi-tape Turing machine depends only on the current configuration, and can be either a change of the current state, a print of a new symbol on each of the cells scanned by the heads, or a move of the heads independently in both directions. Initially, all the tapes are blank except the first one where the input is written. When the machine halts, one recovers the output in the last tape.

Probabilistic Turing machines

A probabilistic (single or multi-tape) Turing machine has an extra tape consisting of symbols forming a support for the uniform distribution. This induces a probability distribution on the outcome corresponding to a given input. In fact, oppositely to a *deterministic* Turing machine, to a given input correspond several computation paths in a probabilistic Turing machine. Therefore, it may well be that for some input x, there are computations which halt and others which don't.

Non-deterministic Turing machines

A non-deterministic Turing machine is a probabilistic Turing machine which accepts strings if at least one computation path, started on these strings, leads to one final state of the machine. Similarly, a language L is accepted (decided) by a non-deterministic Turing machine if the latter accepts all strings $x \in L$.

Oracle Turing machines

An oracle Turing machine is a Turing machine with an extra tape called the oracle tape, and two additional states called the "oracle invocation" and the "oracle appeared" respectively. The configuration of the machine at some time point with state different from the state "oracle invocation" is defined as usual. If a configuration involves the state "oracle invocation" and the string q on the oracle tape, then the next configuration of the machine is identical to the previous one with the exception of moving to the state "oracle appeared" and having on the oracle tape the string r instead. q is called the oracle query and r is the oracle reply. The introduction of such types of machines is motivated by the need to capture the notion of reducibility, which we will see later in this section.

1.1.4 Complexity classes: P, PSPACE, NP, and co-NP

Let DTM be a *deterministic* Turing machine. If for every input word of length n, the machine halts after at most $t(n)$ moves, then TM is said to have a *time complexity* $t(n)$, and the language accepted by DTM is said to be of *time complexity* $t(n)$. The family of languages of deterministic time complexity $O(t(n))$ forms a *complexity class* which we denote $\mathsf{DTIME}(t(n))$. One important complexity class is the class

$$\mathrm{P} = \cup_{k\geq 1}\mathsf{DTIME}(n^k)$$

consisting of languages which can be decided efficiently by a deterministic polynomial Turing machine.

Similarly, if DTM is a deterministic Turing machine that, for every input string of length n, visits at most $s(n)$ cells before halting, then DTM is said to be of *space complexity* $s(n)$, and so is the language accepted by DTM. The family of languages of (deterministic) space complexity $O(s(n))$ forms a complexity class denoted $\mathsf{DSPACE}(s(n))$. The class PSPACE consists of languages that can be decided using a polynomial amount of space .

$$\mathrm{PSPACE} = \cup_{k\geq 1}\mathsf{DSPACE}(n^k).$$

Let now NTM be a *non-deterministic* Turing machine. If for every n-length string, NTM halts after at most $t(n)$ moves, regardless of the selected computational path, then NTM is said to have a time complexity $t(n)$. We define similarly $\mathsf{NTIME}(t(n))$ to be the class of languages that can be decided non-deterministically in time $O(t(n))$. The most important non-deterministic time complexity class is the class

$$\mathrm{NP} = \cup_{k\geq 1}\mathsf{NTIME}(n^k)$$

consisting of languages that can be decided efficiently by a non deterministic polynomial Turing machine.

Finally, to define the class co-NP, one needs to define the complement of a language. According to Subsection 1.1.1, we defined a language L to be the set of positive instances to its underlying

decision problem P. The complement of L, which we denote $\overline{L}$, consists of the negative instances of the problem P. In this way, co-NP is the class comprised of languages whose complements are in NP.

It is not hard to see that P is a subset of NP and co-NP, which are both subsets of PSPACE. The most important question in complexity, which is also called the million dollar question, is to prove/disprove that P is different from NP.

1.1.5 Reductions and completeness

A *reduction* is a transformation of a problem to another. Informally speaking, we say that problem A reduces to problem B if one can solve A given an oracle that solves B. In this case, we say that B is at least as hard as A, and we write $A \leq B$ or $A \Leftarrow B$. Such a reduction is known in the literature as a Turing reduction, where multiple calls to the oracle solving the harder problem are allowed. In case of decision problems, we often use the notion of many-one reduction which corresponds to a Turing reduction where one call to the oracle is allowed. More precisely, a many-one reduction maps an instance x of problem A, to an instance $R(x)$ of the harder problem B such that x is a positive instance of A if and only if $R(x)$ is a positive instance of B. Finally, reductions must be efficient to compute in order to have coherent results. The appropriate notion of efficiency depends on the problems we are studying, for instance, in case of problems/languages in NP, it is convenient to talk about reductions computable in polynomial time.

Let C be a complexity class and L be a given language. We say that L is C*-complete* if $L \in C$, and every language in C is reducible to it. In case $L \notin C$, but still every language in C is reducible to L, we say that L is C-*hard*. Complete problems are important as they are considered to be representatives of the class. In fact, any solution to the complete problem can be used to solve problems in the underlying class. This explains why reductions should be efficient to compute; it would be absurd to have a solution to a complete problem derive a difficult to compute solution to an easier problem.

1.1.6 One way functions and indistinguishability

The bright side of the conjecture $\mathrm{P} \neq \mathrm{NP}$ consists in suggesting different levels of hardness. The most notable ones are the hardness of computing some given values and the hardness of comparing two different entities.

A *one way function* is map which is easy to compute but hard to invert. More precisely, a one way function f is a map from $\Sigma^\star$ to itself, Σ being some alphabet, such that the following holds:

1. for all $x \in \Sigma^\star$, $f(x)$ is at most polynomially longer or shorter than x,

2. there exists a polynomial time Turing machine that, on the input x, outputs $f(x)$.

3. given a uniformly chosen element y from $\Sigma^\star$, there exists no polynomial time Turing machine that returns, with non-negligible probability, x such that $f(x) = y$ if such an x exists, or "no" otherwise.

A *trapdoor* one way function is a one way function such that the knowledge of an additional information (the trapdoor) allows an efficient (polynomial time) computation of its inverse.

Equivalence of two entities varies according to the situations and applications. For example, in some applications, two objects are considered to be equivalent if there exists no *efficient* procedure that differentiates them. This motivates the definition of the different notions of *indistinguishability*. Let p and q be two probability distributions, over some countable probability space $E \subseteq \{0,1\}^n$, that are considered at the security parameter $k \in \mathbb{N}$.

1. p and q are *perfectly indistinguishable* if they are equal.

2. p and q are *statistically indistinguishable* if their statistical difference is negligible in n. We define the statistical difference (or variation distance) of p and q as follows:

$$\Delta(n) = \sum_{e \in E} |p(e) - q(e)|$$

3. p and q are *computationally indistinguishable* if for every probabilistic polynomial time Turing machine M, the following holds:

$$\left| \Pr_{x \leftarrow p}[M(x) = 1] - \Pr_{x \leftarrow q}[M(x) = 1] \right| \leq \frac{1}{n^k}$$

where the expression $x \leftarrow p$ denotes that x has been sampled according to the probability distribution p.

1.1.7 Examples of one way functions

One way functions arise abundantly in cryptography as they offer the possibility of being easy to compute in one way and hard in the other. In fact, this duality easiness/difficulty translates in cryptography into efficiency/security that a cryptographic system should have, since we naturally want the latter to be easy to implement for the honest players but difficult to obstruct by the opponents.

Public key cryptography rests heavily upon two one way functions related to two number theoretic hard problems, namely factoring integers and computing discrete logarithms. Both decision variants of those two problems happen to be in NP, but without being proven to be NP-complete. In fact, one of the main differences between complexity theory and cryptography is that the former considers the worst-case complexity analysis whereas the latter is interested in the average-case analysis. The knapsack problem is one illustration of this difference since it is proven to be NP-complete, however most instances that were used in cryptography have been broken. The upshot

is that one needs an efficient generation of difficult instances of problems in order to be able to use them in cryptography. In this sense, factoring integers and computing discrete logarithms proved to be good candidates for use, which explains the massive design of cryptographic systems based on those problems, or the important mathematic ingenuity spent to solve them.

Factoring

Factoring an integer consists in finding its prime factors. A prime is naturally defined to have only 1 and itself as divisors. Factoring can be easily seen as a one way function since the operation consisting in building an integer from its prime factors, i.e. the multiplication, can be efficiently performed whilst the reverse does not seem to have an efficient algorithmic solution. We summarize in the table below the most known methods to factor an n-bit integer. For a comprehensive and exhaustive list, we refer for example to [Cohen, 1996].

method	**year**	**time**
trial division	$-\infty$	$O(2^{n/2})$
Pollard's $p-1$ method	1974	$O(2^{n/4})$
Pollard's ρ method	1975	$O(2^{n/4})$
Dixon's random squares	1981	$\exp(O(n^{1/2}))$
Lenstra's elliptic curves	1987	$\exp(O(n^{1/2}))$
Number Field Sieve	1991	$\exp(O(n^{1/3}))$

The biggest integer of general form that has been factored so far is the 768-bits RSA challenge using the number field sieve method. An RSA challenge is product of two primes of the same bit size. We refer to [Kleinjung *et al.*, 2010] for details about the factorization of this challenge.

Discrete logarithm

Let $(\mathbb{G}, \cdot)$ be a multiplicative cyclic group generated by some element, say g. The discrete logarithm problem consists in, given an element $y \in \mathbb{G} = \langle g \rangle$, computing x such that $y = g^x$.

It can be easily proven that solving the discrete logarithm problem in a group of order d, with known factorization, can be efficiently reduced to solving the same problem in groups whose orders are the prime factors of d (see for example [Stinson, 2006, Chapter 6]). This explains why we consider in the literature only groups of prime order.

The discrete logarithm problem is proven to be difficult for generic group algorithms. In fact, in [Shoup, 1997], it is proven that algorithms that use no special properties of the considered group need at least $O(\sqrt{d})$ group operations to solve the discrete logarithm problem, where d is the group order. A popular illustration of groups without special properties is given by the group of points of an elliptic curve over a finite field. However, as soon as one considers multiplicative groups of a finite field $\mathbb{Z}_q^\times$ (for a prime q), the cost of solving the discrete logarithm drops drastically to $\exp(O(n^{1/3}))$ using the number field sieve, where n is the bit-size of the considered group order.

On that account, elliptic-curve-cryptography betters finite-field or ring-cryptography as it achieves the same level of security at shorter size parameters. However, we should point that elliptic curves still lose in efficiency when compared to finite fields as it is known that group operations in elliptic curves are much more expensive than their similars in finite fields.

Finally, we note that both factoring and discrete logarithm possess an efficient algorithmic solution using a quantum computer [Shor, 1994]. However, the progress in this area is still not threatening as the largest integer that has been factored so far using this algorithm is 15. Nevertheless, there is a recent trend in cryptography that encourages looking for hard problems that remain hard even in the presence of quantum computers, e.g. lattice or codes-related problems.

1.2 Basic cryptographic primitives

Cryptography was historically associated with the design of systems ensuring confidentiality, namely encryption schemes. However and throughout the years, cryptography evolved to include more systems that serve further purposes. In fact, the digital era gave birth to new applications that require special mechanisms to protect against misuse. Thus, the most appropriate definition of cryptography is, according to [Goldreich, 2001], a science "concerned with construction of schemes that should be able to withstand any abuse. Such schemes are constructed as to maintain a desired functionality, even under malicious attempts aimed at making them deviate from their prescribed functionality".

In this section, we first present the axioms assumed in any cryptographic system, then proceed to a brief description of the most important cryptographic primitives that we will encounter throughout this thesis.

1.2.1 Kerckhoffs' principles

In 1883, Auguste Kerckhoffs formulated in [Kerckhoffs, 1883] the laws or axioms that one should assume about any encryption scheme:

1. The system must be practically, if not mathematically, indecipherable;
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience;
3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents;
4. It must be applicable to telegraphic correspondence;
5. It must be portable, and its usage and function must not require the concourse of several people;

6. Finally, it is necessary, given the circumstances that command its application, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

The most famous law that cryptography owes to Kerckhoffs is stated in the second item, that is, a system should remain secure even if everything except its key is publicly known. This law was reformulated by Claude Shannon by "the enemy knows the system", and later universally adopted in cryptography and in all the subsequent related disciplines, for instance steganography.

1.2.2 Encryption

An encryption scheme is given by the following three algorithms:

Key generation (keygen**).** This is a probabilistic algorithm which returns pairs of encryption and decryption keys $(\mathsf{k}_e, \mathsf{k}_d)$ depending on the security parameter k.

Encryption ($\mathsf{encrypt}$**).** This is a probabilistic algorithm which takes as input an encryption key k_e and a plaintext m, runs on a random tape u and returns a ciphertext c.

Decryption ($\mathsf{decrypt}$**).**This is a deterministic algorithm which takes on input a decryption key k_d, a ciphertext c and returns the corresponding plaintext m or the symbol $\perp$. We require that if $(\mathsf{k}_e, \mathsf{k}_d)$ is a valid key pair, then

$$\forall m\colon \mathsf{decrypt}_{\mathsf{k}_d}\left(\mathsf{encrypt}_{\mathsf{k}_e}(m)\right) = m.$$

Since the antiquity, encryption schemes were conceived such that the keys used for encryption and decryption are the same, which forces the protagonists to meet physically or discuss through a secure channel in order to agree on the key. It is worth noting that such a type of encryption, called symmetric or conventional encryption, was mostly practiced in secret service or military chambers in order to protect state and military communications.

In 1976, Whitfield Diffie and Martin Hellman [Diffie & Hellman, 1976] invented public key encryption, called also asymmetric encryption, where the sender and receiver do not have to agree on the same key to exchange encrypted messages. In fact, the receiver generates a pair of keys k_e and k_d that will be used for encryption and decryption respectively. The receiver will publish the encryption key and store privately the decryption key. With this mechanism, it is obvious that anyone can encrypt a message using k_e, whilst only the receiver can decrypt a ciphertext (obtained using k_e) using the private key k_d. The repercussions of inventing public key cryptography are huge. First it motivated the design of new mechanisms and the introduction of new analysis tools. Then and most importantly, it gave cryptography a scientific shape by allowing more individuals or institutions to participate; cryptography is no longer the workings of some people locked in highly secret military cells, but a production of a whole community that is constantly designing/analyzing systems and publishing the results in well established conferences or journals.

Despite its attractive feature, namely flexibility of the key management, public key encryption did not overwrite symmetric encryption. In fact, the latter compares much better than the former in terms of efficiency. Thus, in most practical applications, both disciplines cohabit under the name "hybrid encryption": public key encryption is first used to communicate a short key, that will be later used to decrypt a huge document, e.g. movie.

Security notions for encryption schemes

Encryption schemes should satisfy a certain security level which clearly identifies the *security goal* the scheme should attain, and the *adversarial power* the attacker against the scheme has. The pair consisting of the security goal and the adversarial power defines what is called the *security notion* for the encryption scheme.

The typical *security goals* a public key encryption scheme should attain are:

1. *Unbreakability (UBK):* it is difficult to recover the private key from the public key of the encryption scheme.

2. *One wayness (OW):* without the private key, it is computationally impossible to recover the plaintext.

3. *Indistinguishability (IND):* the ciphertext reveals no information about the plaintext to a polynomial adversary.

4. *Non-Malleability (NM):* no polynomial adversary can derive from a given ciphertext another ciphertext such that the underlying plaintexts are meaningfully related.

The typical scenario attacks for public key encryption schemes are:

1. *Chosen Plaintext Attack (CPA):* the adversary can encrypt any message of his choice. This is inevitable in public key settings.

2. *Plaintext Checking Attack (PCA):* the adversary is allowed to query an oracle on pairs (m, c) and gets answers whether m is really encrypted in c or not. There is the natural restriction of not querying the oracle on pairs which will help the attacker solving his challenge.

3. *Chosen Ciphertext Attack (CCA):* the adversary is allowed to query a decryption oracle for ciphertexts of his choice. There is again the restriction of not querying the oracle on ciphertexts that will help the attacker solving his challenge.

Remark 1.1. *In the literature, the scenario attack CCA is referred to as CCA2, and is called adaptive chosen ciphertext attack. This is due to the presence of the* non-adaptive chosen ciphertext attack *or the* lunch time attack *scenario, which is denoted CCA1 and where the adversary has the liberty to request the decryption of any ciphertext of his choice up to the challenge phase.*

It is obvious that the CCA attack model is stronger than the PCA attack model which is stronger than the CPA one. We summarize in Figure 1.1 the relations among the different security notions obtained from pairing a security goal GOAL $\in \{\text{OW}, \text{IND}, \text{NM}\}$ and an attack model ATK $\in \{\text{CPA}, \text{PCA}, \text{CCA}\}$. The notation $\mathsf{Notion}_1 \rightarrow \mathsf{Notion}_2$ indicates that if an encryption scheme is secure in the sense Notion_1, then it is also secure in the sense Notion_2; we say that Notion_1 implies Notion_2. The notation $\mathsf{Notion}_1 \leftrightarrow \mathsf{Notion}_2$ means that both Notion_1 and Notion_2 imply each other. Details about the formal definitions of the notions or the proofs underlying Figure 1.1 can be found in [Bellare *et al.*, 1998]. Actually, this work gives also some separation results which we do not report in Figure 1.1 as they are either obtained under some strong assumptions, or they involve notions we do not consider in the thesis (IND $-$ CCA1 or NM $-$ CCA1). Finally, we will provide in Subsection 1.3.1 the formal definitions of the security notions (for public key encryption or for signature schemes) that we will encounter throughout this thesis.

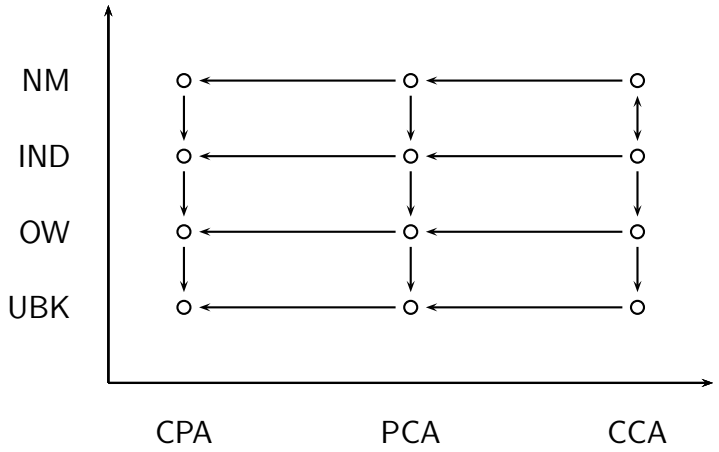

Figure 1.1: Relations among security notions for PKE

Examples of encryption schemes

The most famous public key encryption scheme dates back to 1978 [Rivest *et al.*, 1978]. It is named RSA, which refers to the initials of its inventors, and is depicted in Figure 1.2. The RSA encryption scheme is OW-CPA (one way against a chosen plaintext attack) under the *RSA assumption*, which posits the difficulty of extracting e-th modular roots. However, it is not IND-CPA (indistinguishable against a chosen plaintext attack) since it reveals information about the plaintext, namely $m^e \bmod N$. Less it is NM-CPA (non-malleable against a chosen plaintext attack) as one can compute, given a ciphertext c, another ciphertext, say $c' = 2^e c \bmod N$, whose plaintext m' is meaningfully related to the plaintext m underlying c; $m' = 2m$.

Key generation	Choose two equally-sized primes p, q and compute the modulus $N = pq$, choose $e \xleftarrow{R} \mathbb{Z}^{\times}_{\phi(N)}$, where ϕ is the Euler totient function, compute $d = e^{-1} \bmod \phi(N)$, set the public key pk to (e, N) and the private key sk to (d, N).
Encryption	For a message $m \in \mathbb{Z}^{\times}_N$, compute its encryption as $c = m^e \bmod N$.
Decryption	given a ciphertext c, compute the plaintext as $m = c^d \bmod N$.

Figure 1.2: The RSA encryption scheme

Setup	Choose a group $(\mathbb{G}, \cdot)$ generated by g with prime order d.
Key generation	Choose $x \xleftarrow{R} \mathbb{Z}_d$ and compute $y \leftarrow g^x$, set pk $\leftarrow (d, g, y)$ and sk $\leftarrow (d, g, x)$.
Encryption	For a message $m \in \mathbb{G}$, choose $t \xleftarrow{R} \mathbb{Z}_d$, compute $c_1 \leftarrow g^t$ and $c_2 \leftarrow my^t$, set the ciphertext to (c_1, c_2).
Decryption	Given a ciphertext (c_1, c_2), compute the corresponding plaintext as $m \leftarrow c_2 c_1^{-x}$.

Figure 1.3: The El Gamal encryption scheme

The second famous encryption is due to El Gamal [El Gamal, 1985] and is depicted in Figure 1.3. It was invented in 1985, and it uses the hardness of the discrete logarithm problem. El Gamal's encryption is OW-CPA if the problem, that consists in computing g^{xt} from g^x and g^t, is difficult. Moreover, it is IND-CPA if the problem consisting in distinguishing g^{xt}, given g^x and g^t, from random elements in $\mathbb{G}$, is difficult. We will give in Subsection 1.3.2 a precise definition of these problems.

Encryption with labels Encryption with labels was first introduced in [Shoup & Gennaro, 2002]. In these schemes, the encryption algorithm takes as input, in addition to the public key pk and the message m intended to be encrypted, a label L which specifies information related to the message m and its encryption context. Similarly, the decryption algorithm takes additionally to the ciphertext and private key the label under which the ciphertext was created. Security notions are then defined as usual except that the adversary specifies to his challenger the label to be used in the challenge ciphertext, and in case he (the adversary) is allowed to query oracles, then he cannot query them on the pair formed by the challenge ciphertext and the label used to form it.

1.2.3 Signatures

A signature scheme is given by the following three algorithms:

Key generation (keygen**).** This is a probabilistic algorithm which returns random pairs of private and public keys $(\mathsf{sk}, \mathsf{pk})$ according to the security parameter k.

Signature (sign**).** This is a probabilistic algorithm that takes on input a private key sk and a plaintext m and returns a signature σ.

Verification (verify**).** This is a deterministic algorithm that takes on input a public key pk, a signature σ and outputs 1 if the signature is valid and 0 otherwise. We require that if $(\mathsf{sk}, \mathsf{pk})$ is a valid key pair derived from the algorithm keygen, then for all m, the following holds

$$\mathsf{verify}_{\mathsf{pk}}\left(\mathsf{sign}_{\mathsf{sk}}(m), m\right) = 1.$$

Security notions for signature schemes

Similarly, a signature scheme must meet certain security goals which we list below:

1. *Unbreakability (UBK):* it is difficult to recover the signing key from the verification key.
2. *Universal Unforgeability (UUF):* it is difficult for a polynomial time attacker to obtain a valid signature, without necessarily recovering the private key, on *every* message in the message space.
3. *Selective Unforgeability (SUF):* it is difficult for a polynomial time attacker to produce a valid signature on a message he committed to prior to knowing the public key.
4. *Existential Unforgeability (EUF):* no polynomial time adversary can come up with a valid pair of message and corresponding signature.

It is obvious that existential unforgeability implies universal unforgeability which implies unbreakability.

Moreover, the typical scenario attacks in signature schemes are:

1. *Key Only Attack (KOA):* the adversary has only access to the public key of the scheme, which is unavoidable in the public key scenario.
2. *Known Message Attack (KMA):* the adversary has access to signatures for a set of known messages that he committed to prior to knowing the public key of the scheme.
3. *Chosen Message Attack (CMA):* the adversary can use the signer as an oracle (full access), and may request signatures on any message of his choice.

Remark 1.2. *There exist two further attack scenarios which are weaker than the CMA attack, namely the* Directed Chosen Message Attack (DCMA) *and the* Single Occurrence Chosen-Message Attack (SOCMA). *In the first attack, the adversary chooses non-adaptively a set of messages* $\{m_i\}_i$

and is given the corresponding signatures $\{\sigma_i\}_i$. Whereas in the second attack, the adversary has full access to the signing oracle with the restriction of not querying more than once the same message for signature.

Likewise, pairing the above security goals and the above scenario attacks results in twelve security notions which we describe in Figure 1.4 along with the relations they satisfy. We will provide in Subsection 1.3.1 the formal definitions of the security notions that we will need in this thesis.

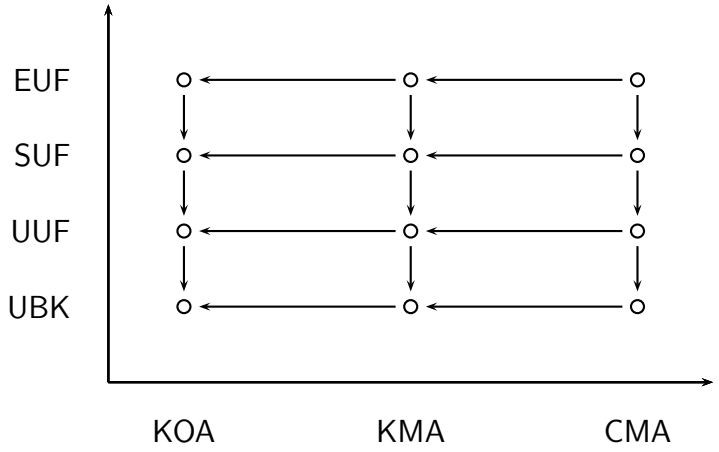

Figure 1.4: Relations among security notion for signature schemes

The RSA signature scheme

One popular signature scheme is the analogous of the RSA encryption scheme which was also described in [Rivest *et al.*, 1978]; we depict it in Figure 1.5. It is obvious that the RSA signature is not existentially unforgeable since one can first choose a signature $s \in_R \mathbb{Z}_N$, then compute its corresponding message as $m = s^e \bmod N$.

1.2.4 Commitment schemes

A commitment scheme [Brassard *et al.*, 1988] consists of the following algorithms:

Setup (setup**).** This is the algorithm that, on input a certain security parameter k, generates the public parameters of the system.

Key generation (keygen**).** This algorithm generates probabilistically a public commitment key pk.

Key generation	Choose two equally-sized primes p, q and compute the modulus $N = pq$, choose $e \xleftarrow{R} \mathbb{Z}^{\times}_{\phi(N)}$, where ϕ is the Euler totient function, compute $d = e^{-1} \bmod \phi(N)$, set the public key pk to (e, N) and the private key sk to (d, N).
Signature	The signature on a message $m \in \mathbb{Z}^{\times}_N$ is computed as $s = m^d \bmod N$.
Verification	For an alleged signature s on m, check whether $m \stackrel{?}{=} s^e \bmod N$.

Figure 1.5: The RSA signature

Commitment (commit**).** This is a probabilistic algorithm that, on input a public key pk and a message m, produces a pair (c, r): c serves as the commitment value (locked box), and r as the opening value.

Opening (open**).** This is a deterministic algorithm that given a pair (c, r) along with a public key pk and an alleged message m, checks whether $(c, r) \stackrel{?}{=} \mathsf{commit}_{\mathsf{pk}}(m)$.

The algorithm open must succeed if the commitment was correctly formed (correctness). Moreover, we require the following security properties:

1. *Hiding.* It is hard for an adversary to generate two messages m_0, m_1 such that he can distinguish between their corresponding locked boxes c_0, c_1. That is, c reveals no information about m. Actually, this notion can be formally described through the following random experiment, for $b \xleftarrow{R} \{0, 1\}$, where $\Omega = (\mathsf{keygen}, \mathsf{commit}, \mathsf{open})$ denotes a commitment scheme with security parameter some $\kappa \in \mathbb{N}$, and $\mathcal{A}$ denotes a PPTM.

   Experiment $\mathbf{Exp}^{\mathsf{hid}-b}_{\Omega,\mathcal{A}}(\kappa)$
   $\mathsf{pk} \leftarrow \Omega.\mathsf{keygen}(\kappa)$,
   $(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}(\mathsf{find}, \mathsf{pk})$
   $(c^\star, r^\star) \leftarrow \Omega.\mathsf{commit}_{\mathsf{pk}}(m_b^\star)$
   $d \leftarrow \mathcal{A}(\mathsf{guess}, \mathcal{I}, c^\star)$
   Return d

   We define the *advantage* of $\mathcal{A}$ *via*:

$$\mathbf{Adv}^{\mathsf{hid}}_{\Omega,\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{hid}-\mathsf{b}}_{\Omega,\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|$$

   Given $t \in \mathbb{N}$ and $\varepsilon \in [0, 1]$, $\mathcal{A}$ is called a (t, ε)-hiding adversary against Ω if, running in time t, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{hid}}_{\Omega,\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme Ω is said to be (t, ε)-hiding if no (t, ε)-hiding adversary against it exists.

Setup	Choose a group $(\mathbb{G}, \cdot)$ generated by g with prime order d.
Key generation	Choose $y \xleftarrow{R} G$ of unknown discrete logarithm w.r.t. g, The public commitment key is y.
Commitment	Compute the commitment (c, r) on a message $m \in Z_d$ such that $r \xleftarrow{R} \mathbb{Z}_d$ and $c = g^r y^m$.
Opening	Given an alleged commitment (c, r) on a message m: check whether $c \stackrel{?}{=} g^r y^m$.

Figure 1.6: Pedersen's commitment scheme

2. *Binding.* It is hard for an adversary to come up with a collision (c, r, r') such that $(c, r) = \mathsf{commit}_{\mathsf{pk}}(m)$ and $(c, r') = \mathsf{commit}_{\mathsf{pk}}(m')$ where pk is a public commitment key, and $m \neq m'$.

3. *Injective.* Given a message m, for any two pairs (c, r) and (c', r') produced using the commit algorithm w.r.t. a public commitment key pk on m such that $r \neq r'$, we have $c \neq c'$.

We call a commitment scheme *secure* if it meets the previous properties.

It is not hard to note the similarity between public key encryption and commitment schemes. In fact, one can check that indistinguishable encryption implies a secure commitment scheme. The main difference between encryption and commitment is that the former requires the decryption algorithm to be based on a "universal" secret key (independent of the message) whilst the latter allows to decrypt with a "message-dependent" secret key, namely the opening value r of the message in question. Another difference is that in encryption, the message is always derived from the ciphertext. This is not always the case in commitments as shows the example depicted in Figure 1.6; it is easy to check that this commitment is correct. Moreover it is statistically hiding because r is random in $\mathbb{Z}_d$ and so is $c = g^r y^m$, regardless of m. Besides, the biding property is achieved under the discrete logarithm assumption in $\mathbb{G}$.

1.2.5 Hash functions

A hash function is used to distill a small amount of information out of large messages. Such an action can ensure integrity of the data in question. In fact, suppose that one maintains a data base in North America and its mirror image in Europe. In order to check that both data bases are identical after for instance an update of both bases, one can compute a so-called *message digest* or *fingerprint* of each data base using the hash function and compare the results; if the data bases are identical then the resulting fingerprints will agree. The converse is not always true since we are mapping a set of large messages to a smaller set of typically 160-bit length strings. However, the event corresponding to having two different data bases mapping to the same fingerprint is very unlikely if the hash function is properly chosen as we will show later in this subsection.

In the rest of this subsection, Σ will denote the set $\{0, 1\}$, whereas n will denote a non-negative integer and $\ell(l)$ an integer such that $\ell(n) > n$. We would like also to note that most material presented in this subsection comes from a course on the topic by Bart Preneel during the summer school "crypt@b-it 2009".

Formal definition

In the discussion above, we considered a fixed hash function, however in more practical situations, it is useful to consider families of hash functions parameterized by keys.

Definition 1.1. *A family of hash functions is a 4-tuple (D, R, K, H) such that:*

1. $D = \Sigma^{\ell(n)}$*, is the set of possible messages, called also the domain of the hash functions family,*
2. $R = \Sigma^n$ *is the finite set of possible fingerprints, called also the range of the hash functions family,*
3. K *is the finite set of possible keys,*
4. H *is the set of hash functions $h_k \in H$, where $k \in K$ and h_k maps messages from D to R.*

Security properties

The most important security properties required in a cryptographic hash function are:

One wayness. Let h be a function with domain $D = \Sigma^{\ell(n)}$ and range $R = \Sigma^n$. h is one-way if it meets the following conditions:

- *Preimage resistance*: let x be selected uniformly in D and let M be an adversary that on the input $h(x)$ outputs, in polynomial time, $M(h(x)) \in D$. For each such an adversary, we require that:
$$\Pr_{x \in_R D}[h(M(h(x))) = h(x)] < \epsilon,$$
where the probability is taken over the input to M as well as on his random tosses, and ϵ is a negligible function in the security parameter.
- *Second preimage resistance*: let x be selected uniformly at random from D and let M be an adversary that on the input $x \in D$ outputs, in polynomial time $x' \in D$ such that $x' \neq x$. For each such an adversary, we require that:
$$\Pr_{x \in_R D}[h(M(h(x))) = h(x)] < \epsilon,$$
where the probability is taken over the input to M as well as on his random tosses, and ϵ is a negligible function in the security parameter.

Collision resistance. Let (D, R, K, H) be a function family with domain $D = \Sigma^{\ell(n)}$ and range $R = \Sigma^n$. Let F be a collision string finder that on input $k \in K$ outputs in polynomial time either ? or a pair $x, x' \in D$ such that $x \neq x'$ and $h_k(x) = h_k(x')$. We require for each such an F the following:

$$\Pr_{k \in_R K}[F(H) \neq \text{"?"}] < \epsilon,$$

where the probability is taken over the random choices of F and of its input $k \in K$.

The work [Rogaway & Shrimpton, 2004] studies the relations (implications and separations) between these properties and further security notions known for hash functions.

Finally, we finish this list with a property required in many cryptographic applications, that is the *random oracle model*, introduced by Bellare and Rogaway in [Bellare & Rogaway, 1996]. In this model, a hash function $h : D \to R$ is chosen uniformly at random from the set of functions from D to R. Moreover, h is not given by a formula or algorithm to compute its outputs. Thus, the only way to compute the value $h(x)$ of some $x \in D$ is through a *call to the function oracle*. This can be assimilated to looking up a huge codebook consisting of values in D and corresponding values in R such that for each possible $x \in D$, there exists a completely random value $h(x) \in R$.

Constructions and issues

The design of cryptographic hash functions started with the iterated structure proposed by Damgård in [Damgård, 1989]. The basic idea of this structure consisted in splitting the message to be hashed into blocks of fixed length, and hashing them block by block with a compression function. The idea was efficient and elegant and has inspired a growing study of the relations between the compression function and the resulting hash function. Moreover, this structure was the origin of two series of celebrated hash functions which are massively used in cryptography that are: MDx (x=4,5) and SHA-y (y=0,1). In fact, the first series of iterated hash functions was due to Rivest and appeared under the name MD4 in 1990, and was later replaced by MD5 due to some weaknesses in the previous version. The next series is called SHA-y (Secure Hash Algorithm) and was conceived by NIST in 1992 (SHA-0) and 1994 (SHA-1). Other constructions of hash functions are based on block ciphers or on algebraic structures, for instance elliptic curves. The advantage of such constructions resides in benefiting from the comprehensive study furnished by their underlying structures, for instance in case of algebraic constructions, one can even come up with formal security proofs, however these constructions remain slow compared to dedicated hash functions.

The current state-of-the-art in hash functions is that all the practical proposals have been broken. Starting from MD4, this algorithm was first shown to have collisions in 1996 by Hans Dobbertin in [Dobbertin, 1996]. A more efficient collision attack was found by the Chinese team of Wang in [Wang *et al.*, 2005]. Generating collisions now in MD4 is as fast as verifying it. MD5 was similarly partially cryptanalyzed by Dobbertin in [Dobbertin, 1996] and later fully broken in [Wang & Yu, 2005] by the same Chinese team. Besides, SHA-0 and SHA-1 had the same fate and

were identified to have weaknesses which argue against keeping them in use. SHA-2 (a set of four hash algorithms, namely SHA-224, SHA-256, SHA-384, and SHA-512)was intact so far however it is algorithmically close to SHA-1 which means that efforts are underway to break it. This has motivated seeking a new hash standard SHA-3 which will be selected via an open competition running between falls 2008 and 2012.

1.2.6 Pseudo random number generators (PRNGs)

Random numbers are of central importance in cryptography. This need manifests for instance when setting up key pairs for cryptographic systems or in probabilistic encryption. Although there seem to be many techniques to obtain random numbers, e.g. system clocks, key strokes or mouse movements etc, most such techniques remain expensive compared to the amount of randomness that needs to be extracted. An illustrative example (from a course on cryptography by Joachim von zur Gathen) is that a 1 GHZ computer running uninterrupted for a year moves through $365 \cdot 24 \cdot 60 \cdot 60 \cdot 10^9$ or $2^{54.8}$ cycles, and thus can only provide 54 random bits (if we take these cycles as random). These bits are certainly not enough for any reasonable protocol, for instance El Gamal's encryption which needs at least a thousand random bits.

To remediate to this problem, cryptographers invented the notion of pseudo random number generators (PRNGs). A PRNG is a deterministic algorithm which inputs strings from a small set X and outputs strings in a larger set Y. The idea consists in starting from a truly random string in X, which would serve as a *seed* for the PRNG, and outputting a string in Y which is indistinguishable from a truly random string in Y. Note that a truly random string in a finite set S is a string which has probability of occurrence $\frac{1}{\#S}$. In Subsection 1.1.6, we discussed many notions of indistinguishability ranging from perfect indistinguishability to computational indistinguishability. In cryptography, as the protagonists are polynomial time algorithms, PRNGs thrive on computational indistinguishability.

PRNGs are proven to exist under the assumption that one way functions exist, and there are many constructions based on any one way function or permutation. We refer to [Goldreich, 2001, Chapter 3] for more details. Finally, PRNGs are massively used in practice and there exists a good number of efficient PRNGs which enjoy a strong security, for instance the Blum Blum Shub PRNG [Blum *et al.*, 1986] based on factoring.

1.3 Reductionist security

We are now able to start a quick browse through a branch of cryptography concerned with gaining confidence on cryptographic schemes, namely reductionist security. In fact, assertions that a system is secure because no one has broken it so far are no longer valid, since experience proved that these systems are broken sooner than later. This is explained by the fact that usually the malicious adversary's view transcends the designer's one. Hence, a new formalism was needed to procure

trust in cryptographic schemes. The following steps have been adopted by designers in the last decade to prove security of their systems:

1. Define clearly the security notion the system needs to meet, by combining the security goal the system should attain and the adversarial power the attacker has access to.

2. Describe a well studied problem $\mathcal{P}$ upon which the security of the system will rest.

3. Provide a *security reduction* from the studied problem to breaking the scheme in question. That is, provide a polynomial time algorithm R that solves the problem $\mathcal{P}$ given access to an algorithm $\mathcal{A}$ breaking the security of the system in the sense defined in Step 1. Such a security proof will guarantee the security of the system if the problem $\mathcal{P}$ is believed to be hard.

Hence, with such a formalism, a system is secure because it captures a high level of security in a strong adversarial model under the reasonable assumption that some well studied problem is hard.

In the rest of this section, we will expand in this topic by defining formally the standard security notions for signature and encryption schemes that will be used later in this thesis. Then, we describe some celebrated assumed "hard" problems. We illustrate afterwards this notion with a small example, and we finish by tackling some advanced topics like idealized proof methodologies or meta-reductions. We wish to note that most material provided in this section comes from two courses on the topic by Pascal Paillier and Marc Joye during the summer schools "crypt@bit 2007" and "crypt@bit 2009" resp.

1.3.1 Notions of security

The standard security notion for digital signatures is the existential unforgeability under a chosen message attack (EUF-CMA), introduced in [Goldwasser *et al.*, 1988]. It is defined through a game between a challenger $\mathcal{R}$ and an adversary $\mathcal{A}$. During this game, $\mathcal{A}$ can obtain signatures on any message of his choice, and at the end, he must output a valid pair message/signature where the output message has not been queried before for signature. The signature scheme is said to be existentially unforgeable if any such an adversary $\mathcal{A}$ has a negligible probability of success in the aforementioned game.

Definition 1.2 (Existential Unforgeability - EUF-CMA)**.** *Let* $\Sigma = (\mathsf{keygen}, \mathsf{sign}, \mathsf{verify})$ *be a digital signature scheme, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment, where* κ *is a security parameter:*

Experiment $\mathbf{Exp}^{\mathsf{euf-cma}}_{\Sigma,\mathcal{A}}(\kappa)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{keygen}(\kappa)$

$(m^\star, \sigma^\star) \leftarrow \mathcal{A}^{\mathfrak{S}}(\mathsf{pk})$

$\mathfrak{S} : m \longmapsto \Sigma.\mathsf{sign}_{\mathsf{sk}}(m)$

return 1 if and only if the following properties are satisfied:

- $\Sigma.\mathsf{verify}_{\mathsf{pk}}[\sigma^\star, m^\star] = \{1\}$
- m *was not queried to* $\mathfrak{S}$

We define the success *of* $\mathcal{A}$ via*:*

$$\mathbf{Succ}^{\textit{euf-cma}}_{\Sigma,\mathcal{A}}(\kappa) = \Pr\left[\mathbf{Exp}^{\textit{euf-cma}}_{\Sigma,\mathcal{A}}(\kappa) = 1\right].$$

Given $(t, q_s) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*,* $\mathcal{A}$ *is said to be a* (t, ε, q_s)*-EUF-CMA adversary against the scheme* Σ *if, running in time* t *and issuing* q_s *signing queries,* $\mathcal{A}$ *has* $\mathbf{Succ}^{\textit{euf-cma}}_{\Sigma,\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* Σ *is called* (t, ε, q_s)*-EUF-CMA secure if no* (t, ε, q_s)*-EUF-CMA adversary against it exists. Finally, we consider a digital signature scheme* Σ *with security parameter* $\kappa \in \mathbb{N}$*;* $\Sigma(\kappa)$ *is said to be EUF-CMA secure if, for any polynomial functions* $t, q_s : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0, 1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_s(\kappa))$*-EUF-CMA secure.*

Remark 1.3. ***(SEUF-CMA)*** *In case* $\mathcal{A}$ *is allowed to output a message already queried to* $\mathfrak{S}$*, yet not with a signature obtained from* $\mathfrak{S}$*, and still does not win the game, the scheme is called SEUF-CMA secure (*S *stands for "strongly").*

In the rest of this subsection, we will define the notions for assymetric encryption that we will encounter later in this thesis, namely NM-CPA, OW-CCA and IND-ATK, for ATK $\in \{$CPA, PCA,-CCA$\}$.

The fist notion that we will present is called non-malleability under a chosen plaintext attack (NM-CPA). It was introduced by Dolev, Dwork, and Naor in 1991 [Dolev *et al.*, 1991], and is defined similarly through a game between a challenger and an adversary $\mathcal{A}$. During this game, $\mathcal{A}$ can only encrypt messages of his choice (inevitable in public key cryptography), and at some point, he outputs to his challenger a distribution D from which messages can be drawn. The challenger picks a message m from D, encrypts it in c and hands it to $\mathcal{A}$. $\mathcal{A}$ continues encrypting messages of his choice, and at the end of the game outputs a binary relation R and a ciphertext c'. $\mathcal{A}$ wins the game if the decryption of c' is related to m via the relation R, and the encryption scheme is proclaimed non-malleable if the success of $\mathcal{A}$ in this game is negligible.

Definition 1.3 (Non-Malleability - NM-CPA)**.** *Let* $\Gamma = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ *be a public key encryption scheme, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment, where* κ *is a security parameter:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{nm-cpa}}_{\Gamma,\mathcal{A}}(\kappa)}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \Gamma.\mathsf{keygen}(\kappa)$

$D \leftarrow \mathcal{A}^{\mathfrak{E}}(\mathsf{pk})$
$\quad \mid \mathfrak{E} : m \longmapsto \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$
$m \xleftarrow{R} D$
$c \leftarrow \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$
$(c', R) \leftarrow \mathcal{A}^{\mathfrak{E}}(\mathsf{pk}, c)$
return (D, R, c, c')

We define the success *of* $\mathcal{A}$ via*:*

$$\mathbf{Succ}^{\textit{nm-cpa}}_{\Gamma,\mathcal{A}}(\kappa) = \Pr[R(m, m')] - \Pr[R(m^\star, m')]$$

where $\mathbf{Exp}^{\textit{nm-cpa}}_{\Gamma,\mathcal{A}}(\kappa) = (D, R, c, c')$*,* $m' = \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c')$ *, and* $m^\star \xleftarrow{R} D$*.*

Given $t \in \mathbb{N}$ *and* $\varepsilon \in [0, 1]$*,* $\mathcal{A}$ *is said to be a* (t, ε)*-NM-CPA adversary against* Γ *if, running in time* t*,* $\mathcal{A}$ *has* $\mathbf{Succ}^{\textit{nm-cpa}}_{\Gamma,\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* Γ *is called* (t, ε)*-NM-CPA secure if no* (t, ε)*-NM-CPA adversary against it exists. Finally, we consider an encryption scheme* Γ *with security parameter* $\kappa \in \mathbb{N}$*;* $\Gamma(\kappa)$ *is said to be NM-CPA secure if, for any polynomial function* $t : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0, 1]$*, it is* $(t(\kappa), \varepsilon(\kappa))$*-NM-CPA secure.*

The next notion that we consider is called one wayness under a chosen ciphertext attack (OW-CCA). One wayness is the oldest and most natural notion public key encryption should satisfy. It was introduced in the seminal work of Diffie and Hellman in [Diffie & Hellman, 1976] to denote the hardness of recovering plaintexts from their corresponding ciphertexts in a given encryption scheme. One wayness under a chosen ciphertext attack refers to the hardness of inverting ciphertexts even in presence of a decryption oracle the adversary can query for any ciphertext except of course on the challenge.

Definition 1.4 (One Wayness - OW-CCA)**.** *Let* $\Gamma = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ *be a public key encryption scheme with message space* $\mathcal{M}$*, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment, where* κ *is a security parameter:*

$$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{ow-cca}}_{\Gamma,\mathcal{A}}(\kappa)}$$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \Gamma.\mathsf{keygen}(\kappa)$
$\mathcal{I} \leftarrow \mathcal{A}^{\mathfrak{D}}(\mathsf{pk})$
$\quad\quad \big| \ \mathfrak{D} : c \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$
$m^\star \xleftarrow{R} \mathcal{M}$
$c^\star \leftarrow \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m^\star)$

$\tilde{m} \leftarrow \mathcal{A}^{\mathfrak{D}}(\mathsf{pk}, c^\star)$
$\quad\quad \big| \ \mathfrak{D} : c\ (\neq c^\star) \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$
return 1 if $\tilde{m} = m^\star$

We define the success *of* $\mathcal{A}$ via*:*

$$\mathbf{Succ}^{\textit{ow-cca}}_{\Gamma,\mathcal{A}}(\kappa) = \Pr\left[\mathbf{Exp}^{\textit{ow-cca}}_{\Gamma,\mathcal{A}}(\kappa) = 1\right].$$

Given $(t, q_d) \in \mathbb{N}^2$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* (t, ε, q_d)*-OW-CCA adversary against* Γ *if, running in time* t *and issuing* q_d *decryption queries,* $\mathcal{A}$ *has* $\mathbf{Succ}^{\textit{ow-cca}}_{\Gamma,\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* Γ *is said to be* (t, ε, q_d)*-OW-CCA secure if no* (t, ε, q_d)*-OW-CCA adversary against it exists. Finally, we consider an encryption scheme* Γ *with security parameter* $\kappa \in \mathbb{N}$*;* $\Gamma(\kappa)$ *is said to be OW-CCA secure if, for any polynomial functions* $t, q_d : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_d(\kappa))$*-OW-CCA secure.*

The last security notion we consider for public key encryption is called indistinguishability or sematic security. It was introduced by Goldwasser and Micali in [Goldwasser & Micali, 1984] and informally denotes the hardness of distinguishing ciphertexts based on their underlying messages. The formal definition of this notion is again through a game between an adversary $\mathcal{A}$ and a challenger. The game runs in three phases; in the first phase, $\mathcal{A}$ has access to the oracles allowed by the given attack model, and eventually outputs two messages $m_0^\star, m_1^\star$ from the message space considered by the given encryption scheme. In the second or challenge phase, the challenger picks uniformly at random one of the messages, encrypts it and gives the result to $\mathcal{A}$. In the last phase, $\mathcal{A}$ continues querying the oracles he had access to in the first phase, which now reject queries made w.r.t. the challenge ciphertext. At the end of the last phase, $\mathcal{A}$ outputs his guess for the message underlying the challenge, and is considered successful if the guess is correct.

Definition 1.5 (Indistinguishability - IND-ATK)**.** *Let* $\Gamma = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ *be a public key encryption scheme with message space* $\mathcal{M}$*, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment, for* $b \xleftarrow{R} \{0,1\}$*, where* κ *is a security parameter:*

Experiment $\mathbf{Exp}^{\textit{ind-atk}-b}_{\Gamma,\mathcal{A}}(\kappa)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \Gamma.\mathsf{keygen}(\kappa)$,
$(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{find}, \mathsf{pk})$

| if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$
| if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} : (m, c) \longmapsto m \stackrel{?}{=} \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$
| if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} : c \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$

$c^\star \leftarrow \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m_b^\star)$
$d \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{guess}, \mathcal{I}, c^\star)$

| if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$
| if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} : (m, c)(\neq (m_i^\star, c^\star), i = 0, 1) \longmapsto m \stackrel{?}{=} \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$
| if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} : c(\neq c^\star) \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}}(c)$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{ind-atk}}_{\Gamma,\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ind-atk-b}}_{\Gamma,\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|$$

Given $(t, q) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*,* $\mathcal{A}$ *is called a* (t, ε, q)*-IND-ATK adversary against* Γ *if, running in time* t *and issuing* q *queries (to the allowed oracle),* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{ind-atk}}_{\Gamma,\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* Γ *is said to be* (t, ε, q)*-IND-ATK secure if no* (t, ε, q)*-IND-ATK adversary against it exists. Finally, we consider an encryption scheme* Γ *with security parameter* $\kappa \in \mathbb{N}$*;* $\Gamma(\kappa)$ *is said to be IND-ATK secure if, for any polynomial functions* $t, q : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0, 1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q(\kappa))$*-IND-ATK secure.*

1.3.2 More hard problems

In Subsection 1.2.2, we presented two encryption schemes, namely RSA [Rivest *et al.*, 1978] and El Gamal [El Gamal, 1985] that are OW-CPA secure if some problems, that are easier than factoring and discrete logarithm respectively, are difficult. In this paragraph, we give a formal description of both problems as well as some of their variants.

RSA-like problems

Definition 1.6. ***The RSA Problem [Rivest* et al.*, 1978].*** *Let* N *be a product of two equally sized primes* p *and* q *(*p *and* q *are* κ*-bit integers). Let further* y *be an integer in* $\mathbb{Z}_N^\times$ *and* $e > 1$ *be an integer co-prime with* $\phi(N)$*. The task of an RSA adversary* $\mathcal{A}$ *is to compute the unique integer* x *in*

$\mathbb{Z}_N^\times$ *such that* $x^e = y \bmod N$. *The advantage of such an adversary is defined by:*

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{r}(p,q,N,e) \leftarrow \mathsf{keygen}(1^\kappa),\\ y \xleftarrow{R} \mathbb{Z}_N^\times,\\ x \xleftarrow{\tau \text{ operations}} \mathcal{A}(N,e,y),\\ x^e = y \bmod N.\end{array}\right]$$

where the probability is taken over the random generation of the RSA instance as well as on all the random choices of the RSA adversary.
Finally, we say that the RSA assumption holds *if we have the following implication:*

$$\tau = \mathsf{poly}(\kappa) \Rightarrow \mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\kappa).$$

Definition 1.7. ***The Flexible RSA Problem [Barić & Pfitzmann, 1997].*** *Let N be a product of two equally sized safe primes, i.e. primes of the form $2p+1$, where p is itself a prime. Let further y denote an integer in $\mathbb{Z}_N^\times$. The task of a Flexible RSA adversary $\mathcal{A}$ is to output an integer $x \in \mathbb{Z}_N^\times$ and an integer $e > 1$ such that $x^e = y \bmod N$. The advantage of such an adversary is defined by:*

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{r}(p',q',N) \leftarrow \mathsf{keygen}(1^\kappa),\\ y \xleftarrow{R} \mathbb{Z}_N^\times,\\ (x,e) \xleftarrow{\tau \text{ operations}} \mathcal{A}(N,y),\\ x^e = y \bmod N \wedge (x,e) \neq (y,1).\end{array}\right]$$

where the probability is taken over the random generation of the Flexible RSA instance as well as on all the random choices of $\mathcal{A}$.
Finally, we say that the Strong RSA (SRSA) assumption holds *if:*

$$\tau = \mathsf{poly}(\kappa) \Rightarrow \mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\kappa).$$

It is easy to see that the RSA problem and its flexible variant are easier than factoring. The reverse is still unclear. Actually, the only results we have about the relation between RSA and factoring are the work [Boneh & Venkatesan, 1998] on the impossibility of reducing *algebraically* factoring to RSA, and the recent proof by Aggarwal and Maurer in [Aggarwal & Maurer, 2009] of the equivalence between factoring and RSA with respect to general ring algorithms.

Diffie-Hellman-like problems

In Subsection 1.2.2, we briefly mentioned that the El Gamal encryption scheme meets different levels of security under the hardness of different problems. We give in the present paragraph a formal description of these problems.

Let $(\mathbb{G} = \langle g \rangle, \cdot)$ be a multiplicative group of order d, generated by g.

Definition 1.8. ***The Computational Diffie-Hellman Problem (CDH).*** *The input to this problem consists of $A = g^a$ and $B = g^b$, where a, b are chosen uniformly at random from $\mathbb{Z}_d$. The adversary $\mathcal{A}$ is then requested to compute a C such that $C = g^{ab}$. The advantage of such an adversary is given by:*

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{r} (G, d, g) \leftarrow \mathsf{keygen}(1^\kappa), \\ (a, b) \xleftarrow{R} \mathbb{Z}_d^\times, \\ g^{ab} \xleftarrow{\tau \text{ operations}} \mathcal{A}(g^a, g^b). \end{array}\right]$$

where the probability is taken over the generation of the CDH instance as well as on the random choices of $\mathcal{A}$.
Similarly, we say that the Computational Diffie-Hellman (CDH) assumption holds *if:*

$$\tau = \mathsf{poly}(\kappa) \Rightarrow \mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\kappa).$$

Definition 1.9. ***The Decisional Diffie-Hellman Problem (DDH).*** *The input to this problem consists of $A = g^a$, $B = g^b$, and $C = g^c$, where a, b are chosen uniformly at random from $\mathbb{Z}_d$ and c is either $ab \bmod d$ or a random element in $\mathbb{Z}_d$. The polynomial time adversary $\mathcal{A}$ is then requested to decide whether $c = ab \bmod d$ or not. Let b be the output of such an adversary, we define its advantage as:*

$$\mathsf{Adv}(\mathcal{A}) = \left|\Pr\left[\begin{array}{r} (G, d, g) \leftarrow \mathsf{keygen}(1^\kappa), \\ (a, b) \xleftarrow{R} \mathbb{Z}_d^\times, \\ b \xleftarrow{R} \{0, 1\} \\ \mathsf{if}\ b = 1\ \mathsf{then}\ \ c \leftarrow ab \bmod d\ \mathsf{else}\ \ c \xleftarrow{R} \mathbb{Z}_d^\times, \\ d \xleftarrow{\tau \text{ operations}} \mathcal{A}(g^a, g^b, g^c), \\ b = d. \end{array}\right] - \frac{1}{2}\right|$$

where the probability is taken over the generation of the DDH instance and on the random choices of $\mathcal{A}$.
Similarly, we say that the Decisional Diffie-Hellman (DDH) assumption holds *if:*

$$\tau = \mathsf{poly}(\kappa) \Rightarrow \mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\kappa).$$

Definition 1.10. ***The Gap Diffie-Hellman Problem (GDH).*** *The input and output of this problem are similar to those of the CDH problem, with the exception of supporting the adversary $\mathcal{A}$ with a DDH oracle that he can query on any DDH instance of his choice.*

$$\mathsf{Adv}(\mathcal{A}) = \Pr\left[\begin{array}{r} (G, d, g) \leftarrow \mathsf{keygen}(1^\kappa), \\ (a, b) \xleftarrow{R} \mathbb{Z}_d^\times, \\ g^{ab} \xleftarrow{\tau \text{ operations}} \mathcal{A}^{\mathsf{DDH}}(g^a, g^b). \end{array}\right]$$

Key generation	Select two primes $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are primes, compute the safe RSA modulus $N = pq$, select a random $s \in_R \mathbb{Z}_N^\times$, consider a collision-resistant hash function $\Psi : \{0,1\}^\ell \to \mathsf{Primes} \geq 3$ ($\ell \approx 30$), set the public key pk to (N, s) and the private key to (p, q).
Signature	A signature on a message $m \in \{0,1\}^\ell$ is computed as $\sigma = s^{1/\Psi(m)} \bmod N$.
Verification	For an alleged signature σ on m, check whether $\sigma^{\Psi(m)} \stackrel{?}{=} s \bmod N$.

Figure 1.7: The GHR signature

where $\mathsf{DDH} : (g^a, g^b, g^c) \longmapsto c \stackrel{?}{=} ab \bmod d$*, and the probability is taken over the generation of the GDH instance and on the random choices of* $\mathcal{A}$*.*
Similarly, we say that the Gap Diffie-Hellman (GDH) assumption holds *if:*

$$\tau = \mathsf{poly}(\kappa) \Rightarrow \mathsf{Adv}(\mathcal{A}) = \mathsf{negl}(\kappa).$$

Remark 1.4. *The CDH, DDH, and GDH problems are* random-self reducible*, i.e. one can generate from a specific instance a random one. Thus, the average case and worst case of all these problems are equivalent.*

Remark 1.5. *The CDH problem is obviously harder than the DDH and GDH problems. There is actually a clear separation between the CDH and the DDH problems in some groups which we will see in Section 1.5.*

We are now able to state that the El Gamal encryption scheme is:

1. OW-CPA secure if the CDH problem is hard, i.e. the CDH assumption holds.
2. IND-CPA secure if the DDH problem is hard, i.e. the DDH assumption holds.
3. OW-PCA secure if the GDH problem is hard, i.e. the GDH assumption holds.

1.3.3 Example: The GHR [Gennaro *et al.*, 1999] signature scheme

We illustrate the principle of reductionist or provable security by one of the simplest security reductions known in the literature: the security proof of the GHR [Gennaro *et al.*, 1999] signature scheme.

Theorem 1.1. *The GHR signature, depicted in Figure 1.7, is EUF-CMA secure if the SRSA assumption holds.*

Proof. Let $\mathcal{R}$ be the Flexible RSA adversary. $\mathcal{R}$ is given the Flexible RSA instance, say (N, y), in addition to an EUF-CMA attacker $\mathcal{A}$, and is requested to come up with a pair (x, e) such that $e > 1$ and $x^e = y \bmod N$. $\mathcal{R}$ needs to generate properly the parameters of the GHR scheme for $\mathcal{A}$ in order to be able to answer the signature queries $\mathcal{A}$ may request. Thus, $\mathcal{R}$ should feed $\mathcal{A}$ with a key $\mathsf{pk} = (N, s)$ such that s allows $\mathcal{R}$ to easily extract $\Psi(m_i)$-th roots of s, where m_i correspond to the messages queried by $\mathcal{A}$ for signature. At the same time, s should be cleverly chosen such that it allows exploitation of the existential forgery output by $\mathcal{A}$ to solve the Flexible RSA instance.

$\mathcal{R}$ will then behave as follows:

Key generation.

- Choose uniformly at random $i \xleftarrow{R} [\![1, 2^\ell]\!]$.
- For each $m_j \in \{0,1\}^\ell$, compute $\Psi(m_j)$ and set $E = \prod_{j \neq i} \Psi(m_j)$.
- Set $s = y^E \bmod N$ and set the GHR public key to (N, s).

Since the provenance of (N, y) is the Flexible RSA instance, thus external to $\mathcal{A}$, and the function $f : y \mapsto y^E$ is one-to-one (E is coprime to $\Phi(N)$), then $(N, s = y^E)$ is perfectly indistinguishable from a random GHR public key.

Signatures simulation. We distinguish two types of messages m_j $\mathcal{A}$ can request for signature:

- either $j \neq i$, in this case $\mathcal{R}$ answers with $y^{E/\Psi(m_j)}$,
- or $j = i$ in which case $\mathcal{R}$ will abort the experiment.

The difference between the simulation provided above and the real execution of the GHR signing algorithm is when $\mathcal{A}$ requests m_i for a signature. Since i is chosen uniformly at random from $[\![1, 2^\ell]\!]$, then the probability that m_i does not belong to the set of queried messages $\{m_{i_1}, \ldots, m_{i_q}\}$ is $\frac{2^\ell - q}{2^\ell} = 1 - \frac{q}{2^\ell}$.

Exploitation of $\mathcal{A}$'s forgery. At some point, $\mathcal{A}$ outputs his forgery σ on $m \notin \{m_{i_1}, \ldots, m_{i_q}\}$, where $\{m_{i_1}, \ldots, m_{i_q}\}$ is the set of messages queried by $\mathcal{A}$. Assume that $m = m_i$, then the forgery satisfies the following equation:

$$\sigma^{\Psi(m_i)} = s = y^E \bmod N.$$

Since E is a product of primes different from the prime $\Psi(m_i)$, then $\mathcal{R}$ can compute integers a and b such that $a \cdot \Psi(m_i) + b \cdot E = 1$. Hence the following holds:

$$y = y^{a\Psi(m_i)} \cdot y^{bE} = y^{a\Psi(m_i)} \sigma^{b\Psi(m_i)} = \left(y^a \sigma^b\right)^{\Psi(m_i)}.$$

$\mathcal{R}$ will then output to his challenger the pair $(x = y^a\sigma^b, e = \Psi(m_i))$. $\mathcal{R}$ solves his RSA challenge if $\mathcal{A}$ produces a forgery on the message m_i. This event occurs with probability $1/(2^\ell - q)$.

Now, if $\epsilon_\mathcal{A}$ is the advantage of the attacker $\mathcal{A}$, then the advantage $\epsilon_\mathcal{R}$ of $\mathcal{R}$ can be computed as:

$$\epsilon_\mathcal{R} = \epsilon_\mathcal{A} \cdot \frac{2^\ell - q}{2^\ell} \cdot \frac{1}{2^\ell - q} = \frac{\epsilon_\mathcal{A}}{2^\ell}.$$

□

Remark 1.6. *The factor 2^ℓ is called the* reduction loss. *A* tight *security reduction is characterized by a small reduction loss. The importance of this factor manifests when we consider* concrete *security instead of* asymptotic *security. In fact, asymptotic security guarantees only that a scheme is asymptotically secure, i.e. all attacks vanish asymptotically if the reduction loss is polynomial in the security parameter and the underlying problem is believed to be asymptotically hard. Concrete security helps to tune the security parameter so that the scheme has a desired concrete security. For example, if the modulus needs to be at least of size 1024 so that the advantage of the Flexible RSA attacker is at most $\epsilon = 2^{-80}$, then with the above reduction, the advantage of the GHR attacker is only smaller than $\epsilon_\mathcal{A} = 2^{-80+\ell} = 2^{-50}$. To have a GHR security about 2^{-80}, one has to increase the size of the modulus.*

Remark 1.7. *There exists a long-message variant of the GHR signature scheme which is proven EUF-CMA secure under the SRSA assumption with a security loss about q, where q is the number of allowed queries. This proof, provided in [Coron, 2002], is shown to be optimal, i.e. there exists no tighter reduction from the Flexible RSA problem to EUF-CMA breaking this variant of GHR.*

1.3.4 Ideal proof models

In Subsection 1.3.3, we provided a security reduction from the Flexible RSA problem to EUF-CMA breaking the GHR signature scheme without making any assumptions on the ingredients of the scheme (group $\mathbb{Z}_N^\times$, the hash function Ψ, etc...); we say that the provided security reduction stands in the *standard model*. Such proofs are usually difficult to obtain even when the design is extremely simple, e.g. RSA-FDH [Bellare & Rogaway, 1996]. This explains why cryptographers resort to *idealizing* some components of the scheme in question and providing a security proof from the presumed hard problem to breaking the scheme with respect to a *generic adversary*, i.e. an adversary accessing the idealized object through an oracle. Such proofs do not provide any insights about the real security of the scheme in the standard model as there exist many designs that are proven secure in idealized settings but insecure in the standard model. However, they provide strong evidence that the scheme in question is secure provided the underlying problem is hard or the adversary does not exploit special properties of the idealized setting.

The popular idealized settings in cryptography are:

The random oracle model (ROM). This is a mathematical abstraction used to model a random hash function. It consists of a theoretical black box that responds to every query with a uniformly chosen random string from the output domain, with the exception of giving the same answer to the same query. A way of simulating the random oracle can be achieved by picking a random element y from the given range for every query x, and storing the pair (x, y) in a history list Hist so that if the same query x is solicited, the reply would be y. Random oracles proved useful in cryptography and they were first considered by Fiat and Shamir in [Fiat & Shamir, 1986] to remove interaction from 3-round public-coin identification schemes. Later, they were used by Bellare and Rogaway in [Bellare & Rogaway, 1993] to provide generic constructions of encryption and signature schemes. As previously mentioned, there are schemes that are proven secure in the ROM but insecure in the standard model. We note for instance the result of Goldwasser and Tauman Kalai [Goldwasser & Tauman Kalai, 2003] that exhibit secure 3-round public-coin identification schemes for which the transformation of Fiat and Shamir in [Fiat & Shamir, 1986] yields insecure digital signature schemes for *any* hash function used in the transformation. This contrasts the work of Pointcheval and Stern [Pointcheval & Stern, 2000] which proved that the Fiat-Shamir methodology always produces EUF-CMA secure digital signatures in the ROM. The result in [Goldwasser & Tauman Kalai, 2003] is strengthened by the work of Paillier and Vergnaud [Paillier & Vergnaud, 2005] which show that some signatures from the Fiat-Shamir paradigm cannot even be UUF-KOA secure in the standard model. Finally, we finish this paragraph by citing a recent positive result about ROM, namely an implementation of a hash function into elliptic curves which is *indifferentiable* from a random oracle. We refer to [Coron & Icart, 2009] for further details.

The generic group model. A generic model of a group was first introduced by Nechaev [Nechaev, 1994]. Shoup [Shoup, 1997] later improved these results and applied this model to cryptography. In this model, one assumes that operations in a group can be performed only by means of an oracle. More specifically, suppose that $\mathbb{G}$ is an (additive) group of prime order q. Then $\mathbb{G}$ is isomorphic to the additive group $\mathbb{Z}_q$ and for any non-identity element $P \in \mathbb{G}$, one can construct an efficient isomorphism sending $i \in \mathbb{Z}_q$ to iP, using some version of the repeated squaring algorithm to perform the scalar multiplication in polynomial time. In a generic group, one assumes that instead of having explicit formulas for the group element iP, we rather have an "encoding" $\sigma(i) \in S \subset \{0,1\}^*$ that represents the element iP. A generic algorithm $\mathcal{A}$ will then consult the oracle for two types of queries:

1. Given an integer $i \in \mathbb{Z}_q$, $\mathcal{A}$ requests the encoding of iP: the oracle will then select randomly a value $\sigma(i)$, to represent the element iP, from the given set of bit strings.
2. Given two encodings $\sigma(i)$ and $\sigma(j)$, $\mathcal{A}$ requests (without knowing necessarily i and j) the encoding of $\sigma(i \pm j)$. Again the oracle responds with a randomly chosen bit-string.

The only condition on the oracle responses is that if the same group element is queried a

second time, the same corresponding encoding must be returned.

One of the important results of this model is the analysis of complexity assumptions in group-based cryptography. For instance, Shoup gave in [Shoup, 1997] lower bounds for solving the discrete logarithm problem and some other related problems. Finally, a security proof in this model assures the absence of an adversary who behaves generically with respect to the given group. However, it does not rule out the existence of a successful adversary for a specific group [Dent, 2002; Stern *et al.*, 2002].

The ideal cipher model. It consists in considering a block cipher as a *random permutation*. A random permutation E takes a pair (k, x) and returns $y = E(k, x)$ which is random in the considered range. Of course $x = E^{-1}(k, y)$. To simulate such a permutation, one proceeds as follows. For any new pair (k, x), pick y at random from the output domain such that $(k, x, y) \notin \mathsf{Hist}[E]$, set $E(k, x) = y$ and return y, and finally update the history $\mathsf{Hist}[E]$ with the record (k, x, y). Such a simulation looks similar to the random oracle model simulation. In fact, equivalence between the ROM and the ideal cipher was left as an open problem until recently where Coron et al. [Coron *et al.*, 2005] showed that security in the ROM implies security in the ideal cipher model; namely they showed that a random oracle can be replaced by a block cipher-based construction, and the resulting scheme remains secure in the ideal cipher model. The other direction was solved three years later in [Coron *et al.*, 2008], however recent works regard the paper in question as incorrect.

1.3.5 Meta-reductions

Meta-reductions are probabilistic oracle (single or multi-oracle) Turing machines, where one oracle tape consists of an efficient reduction from some problem to another. Meta-reductions have been successfully used in a number of important cryptographic results, e.g. the result in [Boneh & Venkatesan, 1998] which proves the impossibility of reducing algebraically factoring to RSA, or the results in [Paillier & Vergnaud, 2005; Paillier, 2007] which show that some well known signatures, which are proven secure in the random oracle, cannot conserve the same security in the standard model. Although most meta-reductions (used in cryptography) apply only to a category of reductions, e.g. key preserving reductions [Paillier & Villar, 2006; Paillier, 2007] or algebraic reductions [Boneh & Venkatesan, 1998; Paillier & Vergnaud, 2005], they constitute an efficient tool to separate cryptographic problems ([Boneh & Venkatesan, 1998]) or to disprove that the security of some cryptographic scheme rests on the hardness of some problem.

Figure 1.8 depicts the typical use of a meta-reduction in disproving that a given problem P reduces to breaking a given signature scheme Σ. Actually, let $\mathcal{R}$ be an algorithm that solves an instance of the problem P, using an attacker $\mathcal{A}$ against the signature scheme. Naturally, $\mathcal{R}$ needs to simulate to $\mathcal{A}$ the key generation, the signature, and the verification algorithms of Σ. If one can build an efficient algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to solve an instance of the same problem P (note that such an algorithm needs to simulate to $\mathcal{R}$ the adversary $\mathcal{A}$), then one can conclude the impossibility

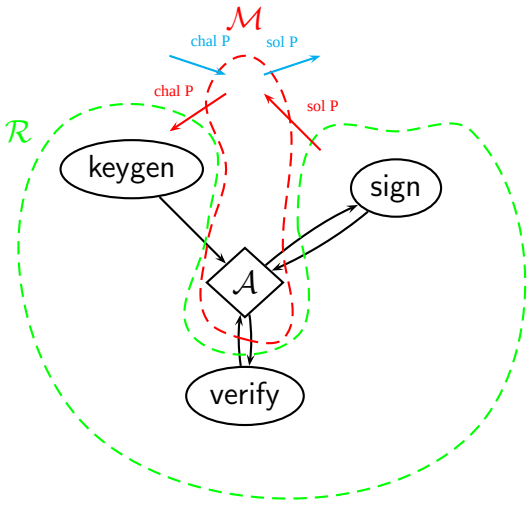

Figure 1.8: Example of a Meta-Reduction

of the existence of $\mathcal{R}$. In fact, existence of $\mathcal{M}$ indicates that under the hardness of P, the algorithm $\mathcal{R}$ does not exist; otherwise, if P is easy, then $\mathcal{R}$ might exist, however its work is useless (solving a problem known to be easy).

1.3.6 Trends in reductionist security

Far from pretending to be exhaustive, this paragraph is confined to shed light on some of the important trends in reductionist security.

Alleviation/removal of idealized models. As previously mentioned, separation results between the standard model and idealized models become more and more popular in cryptography. An interpretation to this is that proofs in these idealized models leave unfair advantage to proofs in the standard model as they modify the adversary's computations in a way that cannot be justified in practice. Thus the need for schemes provably secure in the standard model. There is quite a good number of signature/encryption schemes that are secure in the standard model, however the underlying assumptions are either strong, e.g. [Gennaro *et al.*, 1999; Cramer & Shoup, 2000] or the security reduction is very loose, e.g. [Waters, 2005; Hofheinz & Kiltz, 2009] or the scheme is very inefficient [Hohenberger & Waters, 2009].

Convergence of complexity assumptions. Since the introduction of modern cryptography, many complexity assumptions emerged, most of them were shoveled up from number theory. To name but a few, factoring, RSA, SRSA, discrete logarithm, CDH, DDH, GDH, and many more. A considerable effort in provable security was deployed to study the relation between these assumptions [Maurer & Wolf, 1998; Shoup, 1997; Aggarwal & Maurer, 2009].

Impossibility/optimality proofs. One important issue in security proofs is to spot weaknesses in a design when it fails to exhibit a real attack. In this sense, impossibility proofs aim at showing that a security level cannot be attained by a scheme which suffers some inherent flaws, e.g. [Paillier & Vergnaud, 2005; Paillier & Villar, 2006; Paillier, 2007]. Additionally, optimality proofs, i.e. proofs showing a security reduction to be optimal, gained a lot of popularity in cryptography as the reduction tightness represents an important measure for the concrete security met by a scheme, e.g. [Coron, 2002].

Automatic verification/generation of proofs. Motivated by the tools at the disposal of logicians to verify proofs, e.g. pvs or coq, cryptographers started to check the possibility of automatically verifying and even generating security proofs for their schemes. However, this area remains still unexplored since the only work in this direction is due to Blanchet and Pointcheval [Blanchet & Pointcheval, 2006].

Physical security. So far, the considered security notions defined for cryptographic schemes assume only a *black box* access of the adversary against the scheme to the allowed oracles. This is not very realistic since the adversary might observe the energy consumed by the device while performing the computations, he might also inject faults in these computations. This triggered the crypto community to take into consideration this potential *gray box* access to the oracles in question, and define new security notions accordingly. The state-of-the art in this area is still very modest (see for instance [Goldwasser, 2009] for a survey).

1.4 Zero knowledge (ZK)

A basic problem in cryptography consists of a two-party game where one party tries to prove to the distrustful other party that a statement holds true, without revealing more information other than the validity of the statement in question. We illustrate this situation with the example from [Goldreich, 2001]: suppose that all users in a system keep encrypted backups (using their public keys) of their entire file system in a publicly accessible storage medium. Suppose that at some point, a user Alice wishes to reveal to another user Bob the content of one of her files. One trivial solution consists in decrypting the file in question (using her own private key) and sending it to Bob. The problem with this solution lies in the inability of Bob to check whether the revealed information is really the decryption of the public record. Alice can circumvent the problem by disclosing her private key to Bob, however this will give the latter the possibility of getting hold of her entire file system, which is certainly not desired by Alice.

Such a problem has motivated cryptographers to invent a mechanism allowing Alice to conduct a proof with Bob such that at the end of this proof:

1. Alice is ensured that Bob will not gain any information other than the validity of the statement she tried to prove. Moreover, Bob cannot convince a third party with the validity of the statement in question.

2. Bob accepts the proof only if the statement holds true with a high probability, that is, Alice cannot convince Bob with the validity of an invalid statement.

In this section, we will recall such a mechanism, called zero knowledge proofs of knowledge (ZKPoK). We will first establish the model of computation, namely the model of an interactive proof system, then define the different notions related to this mechanism and that are relevant for the thesis.

1.4.1 Interactive proofs

A model of computation of an interactive proof system was first introduced by Goldwasser, Micali and Rackoff [Goldwasser *et al.*, 1989]. It informally consists of a prover P trying to convince a verifier V that an instance x belongs to a language L. x refers to the common input whereas $(P, V)(x)$ denotes the proof instance carried between P and V at the end of which V is (not) convinced with the membership of the alleged instance x to L:

$$(P, V)(x) \in \{\mathsf{Accept}, \mathsf{Reject}\}$$

P is modeled by a probabilistic Turing machine whereas V is modeled by a *polynomial* probabilistic Turing machine. During $(P, V)(x)$, the parties exchange a sequence of messages called the proof transcript. These messages sizes are polynomial in the size of x. Moreover, $(P, V)(x)$ must terminate in time polynomial in the size of x. The output value $(P, V)(x)$ is a random variable of the common input x, the private input of P and the random coins of both P and V (both P and V are probabilistic Turing machines). We naturally want to have $(P, V)(x) = \mathsf{Accept}$ with high probability for all positive instances ($x \in L$), and with small probability for all negative instances ($x \notin L$). This translates into the following definition (from [Mao, 2008]):

Definition 1.11. *Let L be a language over a given alphabet. We say that a protocol (P, V) is an interactive proof (IP) system for L if:*

$$\Pr\left[(P, V)(x) = \mathsf{Accept} |\ x \in L\right] \geq \epsilon, \tag{1.1}$$

and

$$\Pr\left[(\tilde{P}, V)(x) = \mathsf{Accept} |\ x \notin L\right] \leq \delta \tag{1.2}$$

for every probabilistic Turing machine $\tilde{P}$, where ϵ and δ are constants satisfying

$$\epsilon \in (\frac{1}{2}, 1], \quad \delta \in [0, \frac{1}{2}),$$

where the probability is over all the common input values to (P, V) and all random input values of P, $\tilde{P}$, and V.

Equation 1.1 characterizes the **completeness** notion for an IP protocol, whereas Equation 1.2 characterizes the **soundness** notion which captures the inability of a cheating prover P to convince the verifier V with an invalid statement.

1.4.2 Zero knowledge interactive proofs (ZKIPs)

In the previous subsection, we exhibited a proof mechanism capable of convincing the verifier with the validity of a valid statement. However, we did not address the question of the additional knowledge the verifier will gain aside from the validity of the statement in question. Ideally, we would like this additional knowledge to be *zero*, thus the name *zero knowledge*. We define formally this notion as follows (from [Mao, 2008]):

Definition 1.12. *Let (P, V) be an interactive proof system for some language L. We say that (P, V) is **zero knowledge** if for every $x \in L$, the proof transcript $(P, V)(x)$ can be produced by a probabilistic polynomial-time algorithm (in the size of the input) S with indistinguishable probability distributions:*

- *if the probability distributions of $(P, V)(x)$ and $S(x)$ are the same, then the protocol (P, V) is said to be perfectly zero-knowledge.*
- *if the probability distributions of $(P, V)(x)$ and $S(x)$ are statistically indistinguishable, then (P, V) is called a statistical zero knowledge protocol,*
- *finally, if the distributions of $(P, V)(x)$ and $S(x)$ are computationally indistinguishable, then (P, V) provides only computational zero-knowledgeness.*

Conventionally, the algorithm S is named a simulator for the ZK protocol since it provides a simulation of the proof transcript. However, in case of perfect ZK protocols, S is called often the equator as it provides a perfect simulation.

Remark 1.8 (Honest-verifier zero knowledge (HVZK))**.** *A protocol (P, V) is said to provide only an honest-verifier zero knowledgeness if it is zero knowledge (perfect, statistical or computational) only when the verifier follows honestly the protocol instructions. It may well leak knowledge in the presence of a malicious verifier who does not behave as prescribed. However, it can be shown that every honest-verifier statistical (computational) ZK can be turned efficiently into a statistical (computational) ZK protocol [Goldreich* et al.*, 1998].*

Remark 1.9 (Simulatability of ZKIP)**.** *According to the above definition, a ZKIP assumes the existence of an efficient algorithm capable of producing transcripts indistinguishable from those obtained from the interaction with the real prover. For instance, this simulator is not required to interact with the verifier. However, most ZK (and not only HVZK) proofs in the literature have simulators which interact with the verifier; the idea consists in* rewinding *the verifier until he produces an output that agrees with what the simulator generated beforehand. The example provided later in this section illustrates such a technique which works fine as long as the universe from which the verifier chooses his outputs is polynomially bounded (in the security parameter).*

Finally, throughout this thesis, when we refer to the simulatability of a ZKIP, we mean the existence of a simulator which interacts with the verifier *and produces transcripts that are indistinguishable from those obtained from the interaction with the real prover.*

A complexity theoretic result: NP (co-NP) $\subset$ ZK

An important result in complexity theory shows that every language in NP accepts a zero knowledge proof system. This result has been proven in a constructive manner by first constructing a ZK proof system (P, V) for an NP-complete problem L, e.g. Graph 3-colorability by Goldreich, Micali and Wigderson in [Goldreich *et al.*, 1991] or boolean satisfiability by Brassard, Chaum and Crépeau [Brassard *et al.*, 1988], then propagating this property to the other languages L' in NP as follows:

1. each party computes $x = f(x')$, an instance of the NP-complete language L. It is worth noting that f can by definition be computed and inverted efficiently.

2. P conducts a ZK proof with V to prove that $x \in L$.

It is obvious that the above construction of a ZK proof system for any language in NP constitutes only a theoretic result. In fact, a practical ZK protocol should have the number of interactions between P and V bounded by a linear function in the security parameter. This cannot be achieved by the above construction since we do not know any linear transformation (reduction) of an NP language to an NP-complete one.

Finally, proving that co-NP languages accept also ZK proof systems is done in a more general frame; the above result concerning NP is extended to the class of interactive protocols, namely the class IP, and it is known that this class equals the class PSPACE which contains the class co-NP.

1.4.3 Example of a ZKIP: Schnorr's [Schnorr, 1991] identification protocol

The Schnorr identification protocol was proposed by Schnorr in [Schnorr, 1991] for a real-world (smart card-based) application. This protocol operates in a cyclic group $(\mathbb{G}, \cdot)$ of prime order d which is generated by some element g. The common input of the prover P and verifier V is an element y of unknown discrete logarithm in base g, and the private input of the prover is this very discrete logarithm, say x. That is, P proves to V that he knows x. This protocol is depicted in Figure 1.9. Note that ℓ is a parameter that will be tuned later in the analysis.

Completeness. The completeness of the protocol is trivially achieved with probability $\epsilon = 1$.

Soundness. Suppose that the cheating prover $\tilde{P}$ is able to successfully carry out the above protocol without knowing x. That is, $\tilde{P}$, after having committed to a t, is able to answer the challenge c with a response r satisfying $g^r = ty^c$. Note that, for a fixed t, the last equation corresponds each challenge c to a unique response r. Thus, provided the discrete logarithm problem is hard in $\mathbb{G}$, $\tilde{P}$ needs to guess c correctly beforehand in order to provide an accepting answer; $\tilde{P}$ will first choose $r \stackrel{R}{\leftarrow} \mathbb{Z}_d$, then computes $t = g^r y^{-c}$ and sends it as a commitment in the first step of the protocol. In this way, when $\tilde{P}$ receives the correctly guessed c, he will simply answer with r. This

Prover P		Verifier V
Choose $k \xleftarrow{R} \mathbb{Z}_d$		
Compute $t = g^k$	$\xrightarrow{\quad t \quad}$	
	$\xleftarrow{\quad c \quad}$	Choose $c \xleftarrow{R} \{0,1\}^\ell (c \in \mathbb{N})$
	$\xrightarrow{r = k + cx \bmod d}$	
		Verify that $g^r = ty^c$

Figure 1.9: Proof system for $\{g^a \colon a \in \mathbb{Z}_d\}$ Common input: (y, g) and Private input: $x \colon y = g^x$

results in a soundness error equal to $2^{-\ell}$, which corresponds to the probability of correctly guessing the challenge c. As a consequence, the higher the parameter ℓ, the better for the soundness of the protocol. However, we will see in the next paragraph that we cannot increase this parameter indefinitely since this would compromise the zero knowledgeness of the protocol.

Zero knowledge. For this property, we change sides. We want now to prohibit the verifier from learning anything from the prover apart from the validity of the membership of y to L. For this, we provide the following simulator:

1. Generate uniformly a random challenge $c' \xleftarrow{R} \{0,1\}^\ell$. Choose a random $r \xleftarrow{R} \mathbb{Z}_d$, compute $t = g^r y^{-c'}$, then sends it to the verifier.
2. Get c from the verifier.
3. If $c = c'$, the simulator sends back r. Otherwise, it goes to Step 2 (*rewinds* the verifier).

Let us now analyze the adequacy of this simulator. The prover's first message in the protocol is a random value t in $\mathbb{G}$, and so is the simulator's. Moreover, the distributions of the responses of the prover and of the simulator resp. are again identical. Finally, we observe that the simulator runs in expected time 2^ℓ since the probability of not rewinding the verifier is:

$$\begin{aligned}
\Pr[c = c'] &= \sum_{c_i \in \{0,1\}^\ell} \Pr[c = c_i, c' = c_i] \\
&= \sum_{c_i \in \{0,1\}^\ell} \Pr[c = c_i] \Pr[c' = c_i] \\
&= 2^{-\ell} \sum_{c_i \in \{0,1\}^\ell} \Pr[c = c_i] \\
&= 2^{-\ell}
\end{aligned}$$

Adjusting ℓ to a factor logarithmic in the security parameter ensures that the simulator will run in expected polynomial time.

1.4.4 More on zero knowledge

Since zero knowledge was invented in the mid-eighties, the literature about it was so abundant that it exceeded 400 publications. In this subsection, we will concentrate on the aspects of this notion that are relevant for this thesis.

Further definitions (Σ protocols)

A *public-coin protocol* is a protocol in which the verifier chooses all its messages randomly from publicly known sets. A *three-move protocol* can be written in a canonical form in which the messages sent in the three moves are often called commitment, challenge, and response. The protocol is said to have the *honest-verifier zero-knowledge property (HVZK)* if there exists an algorithm that is able, provided the verifier behaves as prescribed by the protocol, to produce, without the knowledge of the secret, transcripts that are indistinguishable from those of the real protocol. The protocol is said to have the *special soundness property (SSp)* if there exists an algorithm that is able to extract the secret from two accepting transcripts of the protocol with the same commitment and different challenges. Finally, a three-move public-coin protocol with both the HVZK and SSp properties is called a Σ *protocol*.

Round efficiency

As mentioned in the previous subsection, the soundness error in Schnorr's identification protocol amounts to $2^{-\ell}$, where ℓ is a factor logarithmic in the security parameter $\log d$. In order to reduce this error probability to a negligibly small quantity, i.e. a quantity smaller than $1/\log d^c$ for all constants c, we can repeat the protocol $\log d$ many times. Such a protocol is then called a *log-round protocol* which is characterized by a number of rounds linear in the security parameter. There exists also the category of protocols which need to be repeated a polynomial factor (in the security parameter $\log d$) of rounds. We talk then about *poly-round protocols*. Examples of these protocols are for instance those proving the validity of a general NP statement via a general polynomial reduction to a NP-complete problem.

Sequential vs concurrent zero knowledge

We addressed in the previous paragraph the possibility of repeating many times a proof of knowledge in order to reduce its soundness error. This repetition can be sequential or in parallel. The natural question to ask is whether the zero knowledge feature is preserved or not. The good news is that zero knowledge is closed under sequential repetition of the protocol (see [Goldreich, 2001,

Chapter 4, Paragraph 4.3.4] for the proof), which means that we can indefinitely reduce the soundness error of a protocol without compromising its zero knowledgeness. Parallel composition is not however guaranteed to preserve zero knowledge. Less is the concurrent composition which generalizes both sequential and parallel composition; in this composition, many instances of the protocol are invoked at arbitrary times and proceed at arbitrary pace. This composition turns out to be of significant importance in many real life applications. Fortunately, there exists a result [Damgård, 2000] that shows that a wide range of known zero knowledge protocols, e.g. Σ protocols, can be modified with negligible loss of efficiency to preserve zero knowledgeness under concurrent composition.

Non-interactive zero knowledge (NIZK)

This notion, introduced in [Blum *et al.*, 1988], consists of a prover who tries to convince a verifier of the validity of some assertion in one move, i.e. without interaction with the verifier. The basic zero knowledge requirement for such proofs consists in exhibiting an efficient simulator outputting messages indistinguishable from the prover's. It is worth noting here that the definition of the zero knowledge requirement for these proofs is simplified because the verifier cannot affect the prover's actions.

The most famous technique to obtain NIZK from their interactive variants is known as the Fiat-Shamir paradigm [Fiat & Shamir, 1986]. It consists of letting the prover compute the verifier's challenge himself as a hash of the statement to be proved and of the first message. The security of this construction is provided only in the random oracle model, which constitutes its major shortcoming. In fact, it is not in general possible to instantiate the random oracle with a concrete function and have the security properties preserved.

A recent method is due to Damgård et al. [Damgård *et al.*, 2006]. It transforms a 3-move interactive ZK protocol P with linear answer to a non-interactive ZK one (NIZK) using a homomorphic encryption scheme in a registered key model, i.e. in a model where the verifier registers his key. More precisely, let a be the first message computed by the prover in P, $c \in \mathbb{N}$ be the challenge sent by the verifier, and finally let $z = u + cv$ be the answer computed by the prover in the third step, where $u, v \in \mathbb{N}$. Let further Γ denote a homomorphic encryption scheme such that $\Gamma.\mathsf{encrypt}(m + m') = \Gamma.\mathsf{encrypt}(m) \cdot \Gamma.\mathsf{encrypt}(m')$, where m and m' are integer values in a suitable range. If the verifier chooses a key pair $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$ and publishes an encryption e of the challenge c, then the prover can compute a as usual, $\Gamma.\mathsf{encrypt}(z)$ as $\Gamma.\mathsf{encrypt}(u)e^v$, and sends these quantities to the verifier in one pass. The verifier decrypts $\Gamma.\mathsf{encrypt}(z)$ to obtain z and checks whether (a, c, z) is an accepting transcript. The authors in [Damgård *et al.*, 2006] proposed an efficient illustration using Paillier's encryption and the proof of equality of two discrete logarithms.

1.5 Bilinear maps

Bilinear maps are essential in today's cryptography. They are used in constructing many cryptographic schemes ranging from short digital signatures to efficient public key encryption schemes. A bilinear map is nothing but an efficient computable function mapping pairs of group elements to elements in a third group. This function has two properties, namely it is bilinear and it is different from the constant function. More precisely, let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, +)$ be two groups with order d, generated by P and Q respectively. Let $(\mathbb{G}_3, \cdot)$ be another group with the same group order. A bilinear map e is an efficiently computable function from $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ with the following properties:

- Bilinear: $\forall a, b \in \mathbb{Z}_d \colon e(aP, bQ) = e(P, Q)^{ab}$.
- Non-degenerate: $e(P, Q) \neq 1_{\mathbb{G}_3}$.

So far, there seems nothing new since the concept of bilinear functions is already known in mathematics. However, the contribution of cryptographers in this area consists in building efficiently these maps from special and nice groups, i.e. the group of points of an elliptic curve.

In this section, we give a short survey on one popular pairing used in cryptography, namely the Weil pairing. The working of this pairing is not needed in understanding the thesis since bilinear maps are used as black boxes when designing cryptographic schemes. However, we chose to give this short panorama in order to help evaluate the efficiency of systems using such a map. We will first give a short introduction to elliptic curves, then describe how to construct such a pairing.

1.5.1 Introduction to elliptic curves

Let $\mathbb{F}_q$ be a finite field of characteristic $p \geq 5$. A smooth (non-singular) elliptic curve is defined by the Weierstrass equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \;\; a_i \in \mathbb{F}_q$$

or

$$y^2 = x^3 + Ax + B, \;\; A, B \in \mathbb{F}_q, \;\; 4A^3 + 27B^2 \neq 0.$$

We define the group of points of an elliptic curve given by one of the two above equations as follows:

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q \times \mathbb{F}_q \colon y^2 = x^3 + Ax + B\} \cup \{\infty\}$$

The additional point ∞ is called the point at infinity on the elliptic curve. Similarly, we can define $E(\mathbb{F}_{q^k})$, where $\mathbb{F}_{q^k}$ is an extension of the field $\mathbb{F}_q$, by taking the points with coordinates in this extension.

The group operation, which we will denote $+$ in the group $E(\mathbb{F}_q)$, is defined as follows:

- $\forall\, P \in E(\mathbb{F}_q) \colon P + \infty = P$ and $\infty + \infty = \infty$,

- if P and Q are the intersection of E with a vertical line then $P + Q = \infty$,

- otherwise, if $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ then $P + Q = (x_3, y_3)$ such that $x_3 = m^2 - x_1 - x_2$ and $y_3 = m(x_1 - x_3) - y_1$ with

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } x_1 \neq x_2 \\ \frac{3x_1^2 + A}{2y_1} & \text{if } x_1 = x_2 \text{ and } y_1 = y_2 \neq 0 \end{cases}$$

  It is easy to check that with the above definition of the operation $+$, $E(\mathbb{F}_q, +)$ is a finite Abelian group with neutral element ∞. Moreover, we define the order of an element $P \in E(\mathbb{F}_q, +)$ to be the least positive integer m such that $mP = \underbrace{P + P + \cdot + P}_{m \text{ times}} = \infty$.

Definition 1.13 (m-torsion points)**.** *The group of m-torsion points of E is*

$$E[m] = \{P \in E(\overline{\mathbb{F}_q}) \colon mP = \infty\}.$$

Fact 1.2. $E[m] \cong \mathbb{Z}_m \times \mathbb{Z}_m$ *if* $p \nmid m$.

1.5.2 The Weil pairing

The Weil pairing is a map $e \colon E[m] \times E[m] \to \mu_m = \{\zeta \in \overline{\mathbb{F}_q} | \zeta^m = 1\} \subseteq \mathbb{F}_{q^k}^{\times}$, where k is called the embedding degree, which satisfies the following properties:

1. $\forall\, P, Q, R \in E[m] \colon e(P + Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, Q + R) = e(P, Q) \cdot e(P, R)$ (bilinearity).

2. $e(P, Q) = 1\ \forall\, Q \in E[m] \Leftrightarrow P = \infty$ (non-degeneracy)

3. $\forall P \in E[m] \colon e(P, P) = 1$.

The last property of the Weil pairing can be avoided using a distortion map $\Psi : E[m] \to E[m]$ such that P and $\Psi(P)$ belong to disjoint cyclic groups of order m. With this map, we are able to define a modified Weil pairing $\hat{e}$ such that $\hat{e}(P, P) = e(P, \Psi(P))$.

So far, we have presented the most popular pairing in cryptography along with its properties. We will show in the rest of this section how one can efficiently construct such a pairing. We need to first recall the notion of rational functions and their divisors, then proceed to the description of the algorithm computing this pairing.

Rational functions and divisors

A rational function is a ratio of polynomials, e.g.

$$f(x,y) = \frac{f_1(x,y)}{f_2(x,y)}$$

$P = (x, y)$ is called a *zero* of the function f if $f_1(P) = 0$, and is called a *pole* of f if $f_2(P) = 0$.

A rational function $f(x,y) = \frac{f_1(x,y)}{f_2(x,y)}$, where $\hat{f}_1$ and $\hat{f}_2$ denote the top degree parts of f_1 and f_2 respectively, can be evaluated at the special point ∞ as follows

1. if f_1 and f_2 have the same degree, then $f(\infty) = \frac{\hat{f}_1(0,1)}{\hat{f}_2(0,1)}$,
2. if f_1 has larger degree, then $f(\infty) = \frac{\hat{f}_1(0,1)}{0}$,
3. if f_2 has larger degree, then $f(\infty) = \frac{0}{\hat{f}_2(0,1)}$.

Given an elliptic curve E, we can define a rational function on it by simply mapping each of its points $P = (x, y) \in E$ to $f(P) = f(x, y)$. It easy to see that we can write (using the Weierstrass equation that defines the curve E):

$$f(x,y) = u_P(x,y)^r g(x,y)$$

where P is a zero of the rational function f, $r \in \mathbb{Z}$ and P is neither a zero nor a pole of g.
u_P is called a *uniformizer* at P, whereas r is called the *order* of f at P ($r = \mathsf{ord}_P(f)$) which satisfies the following properties:

- if P is neither a zero nor a pole of f, then $\mathsf{ord}_P(f) = 0$,
- if P is zero of f, then $\mathsf{ord}_P(f) > 0$,
- if P is a pole of f, then $\mathsf{ord}_P(f) < 0$.

Finally, a divisor div of a rational function f is defined as follows:

$$\mathsf{div}(f) = \sum_P \mathsf{ord}_P(f)(P),$$

which means that $\mathsf{div}(f)$ evaluates to $\mathsf{ord}(P)$ on the point P. Actually, the notion of a divisor is more general. In fact, a divisor is a map from the points of some curve to the set of integers which is equal to zero except on a finite set of points, called its **support**. To represent this map, it is traditional to write it as a formal sum $\sum D(P)(P)$, where $D(P)$ is the value of the divisor at the point P. The **degree** of a divisor D is simply the (finite) sum of its values at all points. Whereas the **sum** of a divisor $\sum D(P)(P)$ is simply the sum $\sum D(P)P$. Moreover, a divisor is called

principal if it can be written as the divisor of a rational function on the elliptic curve. Finally, if f is an arbitrary function in the function field of an elliptic curve E, and D is an arbitrary divisor of E whose support does not contain any of the zeros or poles of f, then, writing $D = \sum D(P)(P)$, we define:

$$f(D) = \prod f(P)^{D(P)}$$

Fact 1.3. *Any degree zero divisor D can be written as $D = (P) - (\infty) + \mathsf{div}(f)$ for some point P of the elliptic curve and some rational function f.*

Theorem 1.4 (Weil's reciprocity)**.** *Let f and g be two functions in the function field of an elliptic curve. If the zeros and poles of f and g do not intersect, then :*

$$f(\mathsf{div}(g)) = g(\mathsf{div}(f))$$

A proof of this theorem can be found for instance in [Blake *et al.*, 2005, pages 212-213].

Computing the Weil pairing on m-torsion points

We are now able to show how one can compute the Weil pairing on m-torsion points. Let P be an m-torsion point on an elliptic curve E, i.e. $mP = \infty$. To define $e_m(P, Q)$, the Weil pairing for P and Q, we choose two arbitrary divisors D_P and D_Q with distinct support which sum to P and Q respectively. Then we define the two functions f_P and f_Q such that $\mathsf{div}(f_P) = m(P) - m(\infty)$ and $\mathsf{div}(f_Q) = m(Q) - m(\infty)$. $e_m(P, Q)$ is defined as follows:

$$e_m(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)}$$

With this definition, it is easy to check, thanks to Weil's reciprocity law that this map is well defined, i.e. is independent of the choice of D_P and D_Q. Moreover, it is bilinear and non-degenerate. We refer for example to [Joux, 2009b, pages 430-431] for the proof of this claim.

Now, we would like to evaluate the computability of such a map. From the discussion above, it seems mandatory to have an algorithm that efficiently evaluates a function f_P at a point Q. Miller's algorithm [Miller, 2004] (Algorithm 1) does this fairly well. This algorithm considers intermediate functions $f_P^{(i)}$:

$$\mathsf{div}(f_P^{(i)}) = i(P) - (iP) - (i-1)(\infty)$$

with $f_P^{(0)} = f_P^{(1)} = 1$.

It is easy to check that $f_P^{(m)} = f_P$ and that

$$\mathsf{div}(f_P^{(i+j)}) = \mathsf{div}(f_P^{(i)}) + \mathsf{div}(f_P^{(j)}) + (iP) + (jP) - ((i+j)P) - (\infty).$$

It can be shown that there exists a linear polynomial $\mathcal{L}(x, y)$ such that:

$$\mathcal{L}(x, y) = (iP) + (jP) + (-(i + j)P) - 2(\infty).$$

Moreover, if x_0 is the x coordinate of $(i + j)P$, we have:

$$\mathsf{div}(x - x_0) = ((i + j)P) + (-(i + j)P) - 2(\infty).$$

It follows that:

$$\mathsf{div}(f_P^{(i+j)}) = \mathsf{div}(f_P^{(i)}) + \mathsf{div}(f_P^{(j)}) + \mathsf{div}\mathcal{L}(x, y) - \mathsf{div}(x - x_0).$$

As consequence, we can choose:

$$f_P^{(i+j)} = f_P^{(i)} \cdot f_P^{(j)} \cdot \frac{\mathcal{L}(x, y)}{x - x_0}$$

Algorithm 1 Miller's algorithm

Require: An integer $m \geq 0$, m-torsion points P and Q.
Ensure: The value of $f_P(Q)$.

Write m in binary $m = \sum_{i=0}^{k-1} m_i 2^i$
$R \leftarrow P$
$y \leftarrow 1$
for i from $k - 1$ down to 0 **do**
  Let $\mathcal{L}$ be the tangent line at R
  Let $R \leftarrow 2R$
  Let $y \leftarrow y^2 \cdot \mathcal{L}(Q)/(x_Q - x_R)$ in $\mathbb{F}_q$
  if $m_i = 1$ **then**
    $\mathcal{L}$ be the line through P and R
    Let $R \leftarrow R + P$
    Let $y \leftarrow y \cdot \mathcal{L}(Q)/(x_Q - x_R)$ in $\mathbb{F}_q$
  end if
end for
output y

It is easy to see that that Miller's algorithm resembles the repeated squaring algorithm which computes powers of group elements. Optimization of this algorithm can be found in [Cohen & Frey, 2005, pages 417, 424, 432].

Chapter 2

Public Key Encryption Revisited

Abstract. The classical security notion an encryption scheme must fulfill is data privacy or indistinguishability. This notion captures the inability of an attacker to distinguish pairs of ciphertexts based on the messages they encrypt. In [Bellare *et al.*, 2001], the authors propose an additional notion, called anonymity, which formalizes the property of key privacy. As a matter of fact, an adversary, in possession of two public keys and a ciphertext formed by encrypting some data under one of the two keys, should not be able to tell under which key the ciphertext was created. In this chapter, we show that anonymity and indistinguishability are not as orthogonal to each other (i.e. independent) as previously believed. In fact, they are equivalent under certain circumstances. Consequently, we confirm the results of [Bellare *et al.*, 2001] on the anonymity of El Gamal's and of Cramer-Shoup's schemes, based on existing work about their indistinguishability. Finally, we define the notion of anonymity for key and data encapsulation mechanisms (KEMs and DEMs), and we provide a similar study to that of public key encryption on the equivalence between anonymity and indistinguishability for KEMs.
Parts of the results described in this chapter were published in [El Aimani, 2009a] at Africacrypt 2009.

2.1 General framework

The formalization of a security notion capturing key privacy was motivated by the numerous applications in which anonymity surfaced. A typical example is this real-life scenario: a mobile user $\mathcal{A}$ is communicating with a base station $\mathcal{B}$. Assume that an eavesdropper $\mathcal{E}$ knows the set of users communicating with $\mathcal{B}$, and can also listen to the communications of the users with $\mathcal{B}$. In these circumstances, $\mathcal{A}$ still wants to keep his identity (or public key) private from $\mathcal{E}$. This is possible if the ciphertexts do not reveal any information about the public key, namely if the encryption scheme is anonymous.

The formal definition of anonymity for public key encryption was first given in [Bellare *et al.*, 2001]); it is described through a game between a challenger $\mathcal{R}$ and an adversary $\mathcal{A}$. The game runs in three phases. In phase 1, $\mathcal{A}$ is given two challenge public keys pk_0 and pk_1, and has access to the oracles, allowed by the attack model ATK, for both keys pk_0 and pk_1. Once $\mathcal{A}$ decides that phase 1 is over, he outputs to $\mathcal{R}$ a challenge message $m^\star$. In phase 2, $\mathcal{R}$ selects uniformly at random one of the challenge public keys, uses it to encrypt $m^\star$, and hands the resulting ciphertext to $\mathcal{A}$. In phase 3, $\mathcal{A}$ continues querying the oracles he had access to in the first phase, with the restriction of not making queries w.r.t. to the challenge. At the end of phase 3, $\mathcal{A}$ outputs his guess for the public key underlying the challenge ciphertext. $\mathcal{A}$ is considered successful when the output guess is correct.

Definition 2.1 (Anonymity - ANO-ATK)**.** *Let* $\Gamma = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ *be a public key encryption scheme, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment, for* $b \xleftarrow{R} \{0,1\}$*, where* κ *is a security parameter:*

Experiment $\mathbf{Exp}^{\textit{ano-atk}-b}_{\Gamma,\mathcal{A}}(\kappa)$

$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \Gamma.\mathsf{keygen}(\kappa)$,
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \Gamma.\mathsf{keygen}(\kappa)$,
$(m^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{find}, \mathsf{pk}_0, \mathsf{pk}_1)$

- if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$.
- if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0,1; \mathfrak{O}_i : (m,c) \longmapsto m \stackrel{?}{=} \Gamma.\mathsf{decrypt}_{\mathsf{sk}_i}(c)$.
- if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0,1; \mathfrak{O}_i : c \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}_i}(c)$.

$c^\star \leftarrow \Gamma.\mathsf{encrypt}_{\mathsf{pk}_b}(m^\star)$
$d \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{guess}, \mathcal{I}, c^\star)$

- if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$.
- if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0,1; \mathfrak{O}_i : (m,c)(\neq (m^\star, c^\star)) \longmapsto m \stackrel{?}{=} \Gamma.\mathsf{decrypt}_{\mathsf{sk}_i}(c)$.
- if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0,1; \mathfrak{O}_i : c(\neq c^\star) \longmapsto \Gamma.\mathsf{decrypt}_{\mathsf{sk}_i}(c)$.

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{ano-atk}}_{\Gamma,\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ano-atk-b}}_{\Gamma,\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t,q) \in \mathbb{N}^2$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* (t,ε,q)*-ANO-ATK adversary against* Γ *if, running in time* t *and issuing* q *queries to the allowed oracles,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{ano-atk}}_{\Gamma,\mathcal{A}}(\kappa) \geq \varepsilon$*.* Γ *is said to be* (t,ε,q)*-ANO-ATK secure if no* (t,ε,q)*-ANO-ATK adversary against it exists. Finally, we consider an encryption scheme* Γ *with security parameter* $\kappa \in \mathbb{N}$*;* $\Gamma(\kappa)$ *is said to be ANO-ATK secure if, for any polynomial functions* $t, q : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q(\kappa))$*-ANO-ATK secure.*

Exploring the relationship between data privacy and key privacy in public key encryption schemes came very natural to researchers. Indeed, in their seminal work [Bellare *et al.*, 2001], the authors observe that the new notion is totally different from data privacy, as there exist encryption schemes that satisfy one notion but not the other. They also claimed that "it is not hard to see that the goals of data privacy and key privacy are orthogonal". Recently, this claim was proven in [Zhang *et al.*, 2007] by exhibiting a technique that upgrades the key privacy (in an encryption scheme already enjoying this property) but destroys the data privacy, and vice versa. Such a result can be considered as negative, since it only shows how to build a encryption scheme which has one property but not the other. But what about the opposite? Can one specify simple assumptions to hold in an encryption scheme so that key privacy yields data privacy and vice versa? Such an approach has been considered in the literature for a different primitive, namely undeniable signatures. In fact, invisibility and anonymity are two security properties that are closely related in undeniable signatures. The first one requires an adversary not be able to distinguish a valid signature on a certain message from any uniformly chosen bit-string from the signature space, whereas the second notion refers to the hardness of, in possession of a signature and two public keys, telling under which key the signature was created. Since the introduction of undeniable signatures, these two notions were treated separately and many schemes emerged which either meet the first notion or the second, until 2003 where a comprehensive study [Galbraith & Mao, 2003] led to the conclusion that anonymity and invisibility are essentially the same under certain conditions. With such a result, one can seek only one notion when designing undeniable signatures.

In the rest of this chapter, and in an attempt to bridge the gap between anonymity and indistinguishability in encryption schemes, we specify simple conditions to hold in the given encryption scheme so that anonymity implies indistinguishability and vice versa. This will allow a direct use of existing results about data/key privacy of asymmetric encryption schemes rather than "doing the work" from scratch as claimed in [Bellare *et al.*, 2001]. As a consequence, we confirm the results in [Bellare *et al.*, 2001] that prove the anonymity under chosen plaintext attacks of El Gamal's encryption scheme and the anonymity under chosen ciphertext attacks of Cramer-Shoup's encryption, assuming the intractability of the Decisional Diffie-Hellman problem (DDH). Finally, we define the notion of anonymity for key and data encapsulation mechanisms (KEMs and DEMs) and provide a similar study to that of public key encryption on the equivalence between anonymity and indistinguishability for KEMs and DEMs.

2.2 Key privacy vs data privacy

In this section, we present conditions that suffice to conclude on the anonymity of an encryption scheme given existing results about its indistinguishability and vice versa. Our result builds from the work of [Galbraith & Mao, 2003] on undeniable signatures and extends it to public key encryption.

We stress that every choice of the security parameter κ defines a key space $\mathsf{PK} \times \mathsf{SK}$ (corre-

sponding to the space of key pairs $(\mathsf{pk}, \mathsf{sk})$ generated by the keygen algorithm), a message space M and a ciphertext space C. In particular, the ciphertext space C depends merely on κ and not on a specific key.

2.2.1 The main result

Let Γ be a public key encryption scheme given by its three algorithms: $\Gamma.\mathsf{keygen}$, $\Gamma.\mathsf{encrypt}$, and $\Gamma.\mathsf{decrypt}$. The following are the properties needed to prove the relationship between key privacy and data privacy.

Property A: Let κ be a security parameter and let $(\mathsf{pk}, \mathsf{sk})$ be an output of $\Gamma.\mathsf{keygen}$. Consider the uniform distribution on M. Then, the distribution on C corresponding to the random variable $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ ($m \xleftarrow{R} \mathsf{M}$) is computationally indistinguishable from uniform.

Property B: Let κ be a security parameter and let $m \in \mathsf{M}$ be an arbitrary message. Consider the distribution induced by the probabilistic algorithm $\Gamma.\mathsf{keygen}$ on the key space $\mathsf{PK} - \mathsf{SK}$. Then, from a key pair $(\mathsf{pk}, \mathsf{sk})$ sampled according to this distribution, the distribution on C corresponding to the random variable $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ is computationally indistinguishable from uniform.

Intuitively, Property A means basically the following: for a fixed key and varying messages, encryptions look random. It is worth noting that the same property has been formulated differently in [Halevi, 2005], where the author requires the distributions in questions to be statistically indistinguishable. Property B suggests that, for a fixed message and varying keys, encryptions look random.

We get now to the relation between anonymity and indistinguishability. Theorem 2.1 says that if Property A holds in an encryption scheme Γ, then indistinguishability implies anonymity. Theorem 2.2 requires Property B for anonymity to yield indistinguishability in a given encryption scheme. Both theorems stand in all attack models ($\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$).

Theorem 2.1. *Let Γ be a public key encryption scheme that has Property A. Given $(t, q) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$; if Γ is (t, ε, q)-IND-ATK secure, then it is $(t, \frac{\varepsilon}{2}, q)$-ANO-ATK secure, where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$.*

Proof. Given an anonymity adversary $\mathcal{A}^{\mathsf{ano-atk}}$, we will create an indistinguishability adversary $\mathcal{A}^{\mathsf{ind-atk}}$ in the same attack model ATK. Let pk_0 be the input to $\mathcal{A}^{\mathsf{ind-atk}}$. $\mathcal{A}^{\mathsf{ind-atk}}$ will run $\Gamma.\mathsf{keygen}$ to generate a public key pk_1 together with its corresponding private key sk_1.

Queries made by $\mathcal{A}^{\mathsf{ano-atk}}$ are answered in the following way: if they are with respect to the key pk_0, they are forwarded to $\mathcal{A}^{\mathsf{ind-atk}}$'s own challenger. Otherwise, in case they are with respect to pk_1, they are answered by $\mathcal{A}^{\mathsf{ind-atk}}$ using the private key sk_1.

When $\mathcal{A}^{\mathsf{ano-atk}}$ outputs a message m_0 and requests a challenge, $\mathcal{A}^{\mathsf{ind-atk}}$ chooses a message m_1 uniformly at random from M that he will pass, together with m_0 to his challenger. $\mathcal{A}^{\mathsf{ind-atk}}$ will

get an encryption $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_b)$, of either m_0 or m_1 ($b \xleftarrow{R} \{0,1\}$), which he will forward to $\mathcal{A}^{\mathsf{ano-atk}}$. Queries by $\mathcal{A}^{\mathsf{ano-atk}}$ continue to be handled as before.

If $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_b)$ corresponds to the encryption of m_0 (under pk_0), then with overwhelming probability it is not an encryption of m_0 under pk_1. Otherwise, if $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_b)$ is the encryption of m_1 (under pk_0), then by virtue of Property A, $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_1)$ is a random element in C and with overwhelming probability it is not an encryption of m_0 under either key.

At the end of the game, $\mathcal{A}^{\mathsf{ano-atk}}$ outputs a guess b' on the key under which $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_b)$ was created. $\mathcal{A}^{\mathsf{ind-atk}}$ will then output the same guess b'.

Since ϵ is defined to be the advantage of $\mathcal{A}^{\mathsf{ano-atk}}$, we have $\epsilon = |\Pr(b'=0|b=0) - \frac{1}{2}|$. In fact, $\mathcal{A}^{\mathsf{ano-atk}}$ is expected to work only when $b=0$ (proper simulation), which explains the conditional probability. In this case, $\mathcal{A}^{\mathsf{ano-atk}}$ is considered successful when he recognizes the challenge to be an encryption under pk_0 of the message m_0.

The advantage of $\mathcal{A}^{\mathsf{ind-atk}}$ is, according to Definition 1.5, $\left|\Pr(b'=b) - \frac{1}{2}\right|$ and we have:

$$\begin{aligned}
\mathsf{Adv}(\mathcal{A}^{\mathsf{ind-atk}}) &= \left|\Pr(b'=b) - \frac{1}{2}\right| = \left|\Pr(b'=0, b=0) + \Pr(b'=1, b=1) - \frac{1}{2}\right| \\
&= \left|\Pr(b'=0|b=0)\Pr(b=0) + \Pr(b'=1|b=1)\Pr(b=1) - \frac{1}{2}\right| \\
&\approx \left|(\epsilon + \frac{1}{2})\frac{1}{2} + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right| = \frac{1}{2}\epsilon.
\end{aligned}$$

The last inequality, due to $\Pr(b'=1|b=1) \approx \frac{1}{2}$, is explained by the fact that in case $b=1$, there is a negligible chance for $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_0}(m_1)$ to be also an encryption of m_0 under pk_1. □

Theorem 2.2. *Let Γ be a public key encryption scheme that has Property B. Given $(t,q) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$; if Γ is (t,ε,q)-ANO-ATK secure, then it is $(t,\frac{\varepsilon}{2},q)$-IND-ATK secure, where ATK $\in \{CPA, PCA, CCA\}$.*

Proof. From an indistinguishability adversary $\mathcal{A}^{\mathsf{ind-atk}}$ with advantage ϵ, we will construct an anonymity adversary $\mathcal{A}^{\mathsf{ano-atk}}$ as follows.

Let $(\mathsf{pk}_0, \mathsf{pk}_1)$ be the input to $\mathcal{A}^{\mathsf{ano-atk}}$. $\mathcal{A}^{\mathsf{ano-atk}}$ will run $\mathcal{A}^{\mathsf{ind-atk}}$ on pk_0. Queries made by $\mathcal{A}^{\mathsf{ind-atk}}$ will be simply passed to $\mathcal{A}^{\mathsf{ano-atk}}$'s own challenger.

At some time, $\mathcal{A}^{\mathsf{ind-atk}}$ outputs two messages m_0, m_1. $\mathcal{A}^{\mathsf{ano-atk}}$ will forward m_0 to his challenger and obtain the challenge $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_b}(m_0)$ where $b \xleftarrow{R} \{0,1\}$. $\mathcal{A}^{\mathsf{ano-atk}}$ will then pass the challenge to $\mathcal{A}^{\mathsf{ind-atk}}$ and continue to handle queries as previously.

In case $b=0$, the challenge encryption is a valid encryption of m_0 and an invalid encryption of m_1 under pk_0. In the other case, since pk_1 (together with sk_1) is sampled from $\mathsf{PK}-\mathsf{SK}$ and Property B holds, then $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_1}(m_0)$ is a random element in C and with overwhelming probability it is not an encryption of m_1 under pk_0. Therefore, when $\mathcal{A}^{\mathsf{ind-atk}}$ outputs his guess b', $\mathcal{A}^{\mathsf{ano-atk}}$ will forward the same guess to his own challenger.

The advantage of $\mathcal{A}^{\mathsf{ind-atk}}$ in such an attack is defined by: $\epsilon = \left|\Pr(b' = 0|b = 0) - \frac{1}{2}\right|$. In fact, $\mathcal{A}^{\mathsf{ind-atk}}$ is expected to work only when $b = 0$. In this case, $\mathcal{A}^{\mathsf{ind-atk}}$ is considered successful when he recognizes the challenge to be an encryption of m_0 under pk_0.

The overall advantage of $\mathcal{A}^{\mathsf{ano-atk}}$ is according to Definition 2.1:

$$\begin{aligned}
\mathsf{Adv}(\mathcal{A}^{\mathsf{ano-atk}}) &= \left|\Pr(b' = b) - \frac{1}{2}\right| = \left|\Pr(b' = 0, b = 0) + \Pr(b' = 1, b = 1) - \frac{1}{2}\right| \\
&= \left|\Pr(b' = 0'|b = 0)\Pr(b = 0) + \Pr(b' = 1|b = 1)\Pr(b = 1) - \frac{1}{2}\right| \\
&\approx \left|(\epsilon + \frac{1}{2})\frac{1}{2} + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right| = \frac{1}{2}\epsilon.
\end{aligned}$$

In fact, $\Pr(b' = 1|b = 1) \approx \frac{1}{2}$, because in the case where $b = 1$, there is a negligible chance for $\Gamma.\mathsf{encrypt}_{\mathsf{pk}_1}(m_0)$ to be also an encryption of m_1 under pk_0. □

2.2.2 On the orthogonality between key privacy and data privacy

In [Zhang *et al.*, 2007], the authors propose a technique that turns an anonymous encryption scheme into a distinguishable anonymous encryption scheme, and vice versa. The idea consists in considering the augmented scheme which appends the message to its encryption (using the original scheme). Since the new ciphertext does not reveal more information about the public key than the original scheme does, it is still anonymous. Concerning the other part, from an indistinguishable scheme one can consider the encryption scheme consisting of appending the public key to the encryption of the message. The new scheme does not reveal more information about the message than the original scheme does. Therefore, it is still indistinguishable. However, it is not anonymous since it discloses the public key.

Theorem 2.2 complies with this result since the first encryption scheme (obtained by appending the message to the ciphertext) does not have Property B; for a fixed message m, the distribution considered in Property B is easily distinguished from uniform. In fact, the probability that a ciphertext sampled according to this distribution equals a ciphertext whose suffix is different from m is exactly zero. Similarly, Theorem 2.1 is in accordance with this result since the encryption scheme obtained by appending the public key to the ciphertext does not have Property A. Indeed, for a fixed key pk, the probability that a ciphertext sampled from the distribution considered in Property A equals another ciphertext whose suffix differs from pk is exactly zero.

Before concluding this paragraph, it is worth noting that Property A highlights a strength of the discrete-log-based world in contrast to the RSA-based world. Concretely, let Γ be an RSA-based encryption scheme where the public key comprises the RSA modulus N to be used. If the ciphertext c (seen as a set) contains an element $e \in \mathbb{Z}_N$, then the scheme will never have Property A. In fact, for a fixed key pk (where $N \in \mathsf{pk}$) and a message m chosen uniformly at random from M, the probability that $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ equals an element $c' \in \mathsf{C}$ with the component $e' \geq N$,

is exactly zero. Therefore, it is easy to distinguish the distribution on C, defined in Property A, from the uniform distribution. This argument conforms again to the result in [Bellare *et al.*, 2001], namely the fact that RSA-OAEP is not anonymous though it is indistinguishable in the most powerful attack model.

2.3 Application

In the previous paragraph, we showed that our results are consistent with the negative results in [Zhang *et al.*, 2007] concerning the independence of key privacy from data privacy. In fact, as Properties B and A do not hold in the augmented encryption schemes respectively, one cannot deduce one security notion from the other. In this section, we confirm the positive results in [Bellare *et al.*, 2001] concerning the anonymity of El Gamal's [El Gamal, 1985] and of Cramer-Shoup's [Cramer & Shoup, 2003] encryption schemes.

2.3.1 El Gamal's encryption revisited

The ElGamal scheme, described in Figure 1.3 (Subsection 1.2.2), is IND-CPA-secure under the hardness of the Decisional Diffie-Hellman problem (DDH). Actually, the following holds:

$$\mathsf{Adv}(\mathcal{A}^{\mathsf{ind-cpa}}_{\mathsf{ElGamal}}) = \mathsf{Adv}(\mathcal{R}^{\mathsf{ddh}}).$$

To analyze the ANO-CPA property of El Gamal, it suffices to check whether Property A holds. The ciphertext space C consists of:

$$\mathsf{C} \;=\; \Big\{(g^t, my^t) \in \mathbb{G}\times\mathbb{G}\colon t \xleftarrow{R} \mathbb{Z}_d,\ m \in \mathsf{M},\ (y=g^x, x) \in \mathsf{PK}-\mathsf{SK}\Big\} = \mathbb{G}\times\mathbb{G}.$$

We show now that the distribution on C, corresponding to the random variable $\mathsf{ElGamal.encrypt}_y(m)$, where y is a fixed public key and m is a message sampled uniformly at random from M, is exactly the uniform distribution. Let $(a_1, a_2) \in \mathsf{C}$ be a fixed value from $\mathbb{G}\times\mathbb{G}$.

$$\begin{aligned}\Pr[(g^t, my^t) = (a_1,a_2)] &= \Pr[g^t = a_1]\Pr[my^t = a_2 | g^t = a_1] \\ &= \frac{1}{d}\Pr[my^t = a_2 | y^t = a_1^x] = \frac{1}{d}\Pr[m = a_2 a_1^{-x}] = \frac{1}{d^2}.\end{aligned}$$

The last equality is due to the fact that m was sampled uniformly at random from $\mathsf{M} = \mathbb{G}$. We conclude with Theorem 2.1 that El Gamal's encryption is ANO-CPA secure under the DDH assumption and we have: $\mathsf{Adv}(\mathcal{R}^{\mathsf{ddh}}) \approx \frac{1}{2}\mathsf{Adv}(\mathcal{A}^{\mathsf{ano-cpa}}_{\mathsf{ElGamal}})$, which complies with Theorem 1 in [Bellare *et al.*, 2001].

2.3.2 Cramer-Shoup's encryption revisited

Cramer-Shoup's encryption scheme [Cramer & Shoup, 2003] is IND-CCA secure under the DDH assumption. It uses a prime order group $(\mathbb{G}, \cdot)$ with order d, and given by two generators g_1 and g_2. Furthermore, it requires a family of collision resistant hash functions $\mathcal{H} = (\mathcal{HG}, \mathcal{HE})$, defined by a probabilistic generator algorithm $\mathcal{HG}$ - which takes as input the security parameter κ and returns a key K - and a deterministic algorithm $\mathcal{HE}$ - which takes as input the key K and a string $m \in \{0,1\}^*$ and returns an element in $\mathbb{Z}_d$:

$\mathsf{setup}(\kappa)$	keygen	$\mathsf{encrypt}_{\mathsf{pk}}(m)$	$\mathsf{decrypt}_{\mathsf{sk}}(u_1, u_2, e, v)$
$(d, g_1) \xleftarrow{R} \bar{\mathcal{G}}$	$x_1, x_2, y_1, y_2, z \xleftarrow{R} \mathbb{Z}_d$	$r \xleftarrow{R} \mathbb{Z}_d$	$\alpha \leftarrow \mathcal{EH}(u_1, u_2, e)$
$g_2 \xleftarrow{R} \mathbb{G}_d$	$c \leftarrow g_1^{x_1} g_2^{x_2}; d \leftarrow g_1^{y_1} g_2^{y_2}$	$u_1 \leftarrow g_1^r; u_2 \leftarrow g_2^r$	if $u_1^{x_1+\alpha y_1} u_2^{x_2+\alpha y_2} = v$
$K \leftarrow \mathcal{HG}(\kappa)$	$h \leftarrow g_1^z$	$e \leftarrow mh^r$	then $m \leftarrow e u_1^{-z}$
$\mathsf{Return}(d, g_1, g_2, K)$	$\mathsf{pk} \leftarrow (d, g_1, g_2, c, d, h, K)$	$\alpha \leftarrow \mathcal{HE}_K(u_1, u_2, e)$	else $m \leftarrow \perp$
	$\mathsf{sk} \leftarrow (x_1, x_2, y_1, y_2, z)$	$v \leftarrow c^r d^{r\alpha}$	$\mathsf{Return}(m)$
	$\mathsf{Return}(\mathsf{pk}, \mathsf{sk})$	$\mathsf{Return}(u_1, u_2, e, v)$	

To analyze the anonymity of the scheme, it suffices to check Property A. We have:

$$\mathsf{C} = \left\{ (g_1^r, g_2^r, mh^r, c^r d^{\alpha r}) \colon r \xleftarrow{R} \mathbb{Z}_d, (m, c, d, h) \in \mathsf{M} \times \mathsf{PK}, \alpha = \mathcal{EG}(g_1^r, g_2^r, mh^r) \right\}.$$

It is then easy to see that the size of C is d^3. Therefore, to show that Property A holds, it suffices to show that for a fixed key $\mathsf{pk} = (c, d, h)$ [1] and a message $m \xleftarrow{R} \mathsf{M}$, the probability that $\mathsf{encrypt}_{\mathsf{pk}}(m) = (g_1^r, g_2^r, mh^r, c^r d^{\alpha r})$ equals a given value $(a_1, a_2, a_3, a_4) \in \mathsf{C}$ is exactly $\frac{1}{d^3}$:

$$\begin{aligned}
\Pr[(g_1^r, g_2^r, mh^r, c^r d^{\alpha r}) &= (a_1, a_2, a_3, a_4)] \\
&= \Pr[g_1^r = a_1] \Pr[g_2^r = a_2 | g_1^r = a_1] \cdot \\
&\quad \Pr[mh^r = a_3 | (g_1^r, g_2^r) = (a_1, a_2)] \cdot \\
&\quad \Pr[c^r d^{\alpha r} = a_4 | (g_1^r, g_2^r, mh^r) = (a_1, a_2, a_3)] \\
&= \frac{1}{d} \cdot \Pr[\mathsf{DL}_{g_1}(a_1) = \mathsf{DL}_{g_2}^{a_2}] \cdot \Pr[m = a_3 a_1^{-z}] \cdot \\
&\quad \Pr[\mathcal{EH}_K(a_1, a_2, a_3) = \mathsf{DL}_{a_1^{y_1} a_2^{y_2}}(a_4 a_1^{-x_1} a_2^{-x_2})] \\
&= \frac{1}{d} \cdot 1 \cdot \frac{1}{d} \cdot \frac{1}{d} = \frac{1}{d^3}.
\end{aligned}$$

In fact, since $(a_1, a_2, a_3, a_4) \in \mathsf{C}$, then $\mathsf{DL}_{g_1}(a_1) = \mathsf{DL}_{g_2}(a_2)$ holds with probability 1. Moreover, as m was chosen uniformly at random from $\mathsf{M} = \mathbb{G}$, then the probability that m equals a given value in $\mathbb{G}$ is exactly $\frac{1}{d}$. Finally, the relationship ($\mathcal{EH}_K$ is a deterministic algorithm)

[1] Note that g_1, g_2 and K are fixed for all keys in the setup algorithm.

$\mathcal{EH}_K(a_1, a_2, a_3) = \mathsf{DL}_{a_1^{y_1} a_2^{y_2}}(a_4 a_1^{-x_1} a_2^{-x_2})$ holds in $\mathbb{Z}_d$, for arbitrary values $(a_1, a_2, a_3, a_4) \in \mathsf{C}$ with probability $\frac{1}{d}$. We conclude with Theorem 2.1 that Cramer-Shoup's encryption is ANO-CCA-secure under the DDH assumption.

2.4 Key and data encapsulation mechanisms (KEMs & DEMs)

Key and data encapsulation mechanisms arise very often in cryptography. In fact, they are both combined to build public key encryption schemes using the so-called "hybrid encryption paradigm"; a KEM is first used to fix a *session key* along with its *encapsulation*, then the DEM (which is nothing but a secret key encryption algorithm) is used to encrypt the message in question using the session key. Decryption is achieved by first recovering the key from the encapsulation (part of the ciphertext) then applying the DEM decryption algorithm using the recovered key.

In this section, we recall the formal definition of KEMs and DEMs, then we define the anonymity security notion for these mechanisms, and we provide a study of the equivalence between this new notion and the traditional indistinguishability notion.

2.4.1 Key encapsulation mechanisms (KEMs)

A KEM is a tuple which comprises the following algorithms:

1. **Setup (**setup**).** This algorithm generates the public parameters of the scheme.

2. **Key generation (**keygen**).** This algorithm probabilistically generates, on input a security parameter κ, a key pair $(\mathsf{pk}, \mathsf{sk})$.

3. **Encapsulation (**encap**).** This algorithm inputs the public key pk, runs on a random tape u, and generates a *session key* denoted k and its *encapsulation* c.

4. **Decapsulation** decap**.** Given the private key sk and the element c, this algorithm computes the decapsulation k of c, or returns $\perp$ if c is invalid.

The standard security goal for KEMs is indistinguishability. It informally means the hardness of distinguishing the key corresponding to an arbitrary encapsulation from a uniformally chosen bit-string from the (session) key space. We give below the formal definition of this property.

Definition 2.2 (Indistinguishability (KEMs) - IND-ATK)**.** *Let $\mathcal{K} = (\mathsf{keygen}, \mathsf{encap}, \mathsf{decap})$ be a KEM with session key space K, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, for $b \xleftarrow{R} \{0,1\}$, where κ is a security parameter:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\textit{ind-atk-b}}_{\mathcal{K},\mathcal{A}}(\kappa)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{K}.\mathsf{keygen}(\kappa)$,

$\mathcal{I} \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{find}, \mathsf{pk})$

- if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$
- if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} : (c, k) \longmapsto k \stackrel{?}{=} \mathcal{K}.\mathsf{decap}_{\mathsf{sk}}(c)$
- if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} : k \longmapsto \mathcal{K}.\mathsf{decap}_{\mathsf{sk}}(c)$

$(c^\star, k^\star) \leftarrow \mathsf{k}.\mathsf{encap}_{\mathsf{pk}}()$

if $b = 0$ then $\{k \stackrel{R}{\leftarrow} \mathsf{K}, k^\star \leftarrow k\}$

$d \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{guess}, \mathcal{I}, c^\star, k^\star)$

- if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$
- if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} : (c, k)\ (\neq (c^\star, k^\star)) \longmapsto k \stackrel{?}{=} \mathcal{K}.\mathsf{decap}_{\mathsf{sk}}(k)$
- if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} : c\ (\neq c^\star) \longmapsto \mathcal{K}.\mathsf{decap}_{\mathsf{sk}}(c)$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{ind-atk}}_{\mathcal{K},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ind-atk-b}}_{\mathsf{k},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*,* $\mathcal{A}$ *is called a* (t, ε, q)*-IND-ATK adversary against* $\mathcal{K}$ *if, running in time* t *and issuing* q *queries to the allowed oracles,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{ind-atk}}_{\mathcal{K},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* $\mathcal{K}$ *is said to be* (t, ε, q)*-IND-ATK secure if no* (t, ε, q)*-IND-ATK adversary* $\mathcal{A}$ *against it exists. Finally, we consider a KEM* $\mathcal{K}$ *with security parameter* $\kappa \in \mathbb{N}$*;* $\mathcal{K}(\kappa)$ *is said to be IND-ATK secure if, for any polynomial functions* $t,\ q : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0, 1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q(\kappa))$*-IND-ATK secure.*

An example of a KEM is the mechanism underlying the El Gamal encryption ($(\mathbb{G}, \cdot)$ is a group generated by g where $|\mathbb{G}| = d$):

Example 2.3. *The most famous and probably oldest KEM known in the literature is the mechanism underlying El Gamal's encryption [El Gamal, 1985]. We depict this KEM in Figure 2.1. The El Gamal KEM is trivially IND-CPA secure under the DDH assumption.*

Example 2.4. *Another popular KEM was introduced in [Boneh* et al.*, 2004a], and is titled the Linear Diffie-Hellman KEM. We depict this KEM in Figure 2.2. The Linear Diffie Hellman KEM is IND-CPA secure under the hardness of* decision linear problem*, which we describe in Definition 2.5.*

Definition 2.5 (Decision Linear Problem (DLP))**.** *Given* $U,\ V,\ H,\ aU,\ bV,\ cH \in \mathbb{G}$*, output* 1 *if* $a + b = c \bmod (\#\mathbb{G})$ *and* 0 *otherwise.*

Setup	Consider a group $(\mathbb{G}, \cdot)$, generated by g where $\lvert\mathbb{G}\rvert = d$.
Key generation	Choose $x \xleftarrow{R} \mathbb{Z}_d$ and compute $y \leftarrow g^x$, set $\mathsf{pk} \leftarrow (d, g, y)$ and $\mathsf{sk} \leftarrow (d, g, x)$.
Encapsulation	Choose $t \xleftarrow{R} \mathbb{Z}_d$ and compute g^t and y^t, set the session key $k \leftarrow y^t$ and its encapsulation $c \leftarrow g^t$.
Decapsulation	One recovers the key y^t from g^t as follows $y^t \leftarrow (g^t)^x$.

Figure 2.1: The El Gamal KEM

Setup	Consider a bilinear additive group $(\mathbb{G}, +)$, with prime order d, generated by P.
Key generation	Generate two secret values $x_1, x_2 \xleftarrow{R} \mathbb{Z}_d^{\times}$, and compute $X_1 \leftarrow x_1 P$ and $X_2 \leftarrow x_2 P$, set the private key $\mathsf{sk} \leftarrow (x_1, x_2)$ and the public key $\mathsf{pk} \leftarrow (X_1, X_2)$.
Encapsulation	Choose a random nonce $(a, b) \xleftarrow{R} \mathbb{Z}_d^2$, generate the session key $k \leftarrow (a + b)P$ and its encapsulation $c \leftarrow (aX_1, bX_2)$.
Decapsulation	Given the private key sk and the encapsulation $c = (aX_1, bX_2)$, compute the key k as $k \leftarrow x_1^{-1} a X_1 + x_2^{-1} b X_2$.

Figure 2.2: The Linear Diffie-Hellman KEM

Anonymity in KEMs

We define similarly anonymity for KEMs to be the hardness of distinguishing pairs of encapsulations/keys based on the underlying public key. Combining this goal with the different attack models $\{\text{CPA}, \text{PCA}, \text{CCA}\}$ results in three security notions which we formally present as follows:

Definition 2.6 (Anonymity (KEMs) - ANO-ATK)**.** *Let $\mathcal{K} = (\mathsf{keygen}, \mathsf{encap}, \mathsf{decap})$ be a KEM, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, for $b \xleftarrow{R} \{0, 1\}$, where κ is a security parameter:*

Experiment $\mathbf{Exp}^{\textit{ano-atk}-b}_{\mathcal{K},\mathcal{A}}(\kappa)$

$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathcal{K}.\mathsf{keygen}(\kappa)$,
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathcal{K}.\mathsf{keygen}(\kappa)$,
$\mathcal{I} \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{find}, \mathsf{pk})$

| if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$.
| if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0, 1; \mathfrak{O}_i : (c, k) \longmapsto k \stackrel{?}{=} \mathcal{K}.\mathsf{decap}_{\mathsf{sk}_i}(c)$.
| if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0, 1; \mathfrak{O}_i : k \longmapsto \mathcal{K}.\mathsf{decap}_{\mathsf{sk}_i}(c)$.

$(c^\star, k^\star) \leftarrow \mathcal{K}.\mathsf{encap}_{\mathsf{pk}_b}()$
$d \leftarrow \mathcal{A}^{\mathfrak{O}}(\textit{guess}, \mathcal{I}, c^\star, k^\star)$

| if $\mathsf{atk} = \mathsf{cpa}$ then $\mathfrak{O} = \mathsf{empty}$.
| if $\mathsf{atk} = \mathsf{pca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0, 1; \mathfrak{O}_i : (c, k)\ (\neq (c^\star, k^\star)) \longmapsto k \stackrel{?}{=} \mathcal{K}.\mathsf{decap}_{\mathsf{sk}_i}(c)$.
| if $\mathsf{atk} = \mathsf{cca}$ then $\mathfrak{O} = \mathfrak{O}_i, i = 0, 1; \mathfrak{O}_i : c\ (\neq c^\star) \longmapsto \mathcal{K}.\mathsf{decap}_{\mathsf{sk}_i}(c)$.

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{ano-atk}}_{\mathcal{K},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ano-atk-b}}_{\mathcal{K},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*,* $\mathcal{A}$ *is called a* (t, ε, q)*-ANO-ATK adversary against* $\mathcal{K}$ *if, running in time* t *and issuing* q *queries to the allowed oracles,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{ano-atk}}_{\mathcal{K},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* $\mathcal{K}$ *is said to be* (t, ε, q)*-ANO-ATK secure if no* (t, ε, q)*-ANO-ATK adversary against it exists. Finally, we consider a KEM* $\mathcal{K}$ *with security parameter* $\kappa \in \mathbb{N}$*;* $\kappa(\kappa)$ *is said to be ANO-ATK secure if, for any polynomial functions* $t,\ q : \mathbb{N} \to \mathbb{N}$*, and any non-negligible function* $\varepsilon : \mathbb{N} \to [0, 1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q(\kappa))$*-ANO-ATK secure.*

Similarly to the study provided in the previous section, we formulate a further property which is sufficient for anonymity to induce indistinguishability. Informally speaking, this property suggests that for a fixed encapsulation c and varying public keys pk (with the corresponding private keys sk), the resulting decapsulations $\mathsf{decap}_{\mathsf{sk}}(c)$ look random.

Again, we stress that every choice of the security parameter κ defines a key space $\mathsf{PK} \times \mathsf{SK}$ (corresponding to the space of key pairs $(\mathsf{pk}, \mathsf{sk})$), an encapsulation space C (corresponding to the encapsulations generated by the KEM encapsulation algorithm) and a session key space K (corresponding to the session keys generated by the KEM decapsulation algorithm).

Property C: Let κ be a security parameter. Let further c be an arbitrary encapsulation value from C. Consider the distribution induced by the probabilistic algorithm keygen on the key space $\mathsf{PK} \times \mathsf{SK}$. Then, from a key $(\mathsf{pk}, \mathsf{sk})$ sampled according to this distribution, the distribution on K, corresponding to the random variable $\mathsf{decap}_{\mathsf{sk}}(c)$, is computationally indistinguishable from uniform.

Remark 2.1. *The KEM underlying the El Gamal encryption scheme satisfies trivially this property, and so does the linear Diffie-Hellman KEM.*

Note that there exist evidently KEMs which do not fulfill this property, for instance KEMs where the decapsulation algorithm returns $\perp$ for some keys; for such KEMs, we cannot use Theorem 2.3 to derive indistinguishability from anonymity.

Theorem 2.3. *Let $\mathcal{K}$ be a key encapsulation mechanism that has Property C. Given $(t, q) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$; if $\mathcal{K}$ is (t, ε, q)-ANO-ATK secure, then it is (t, ε, q)-IND-ATK secure, where ATK $\in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$.*

Proof. First assume that the distribution on the session keys space K (considered in Property C) is *exactly* the uniform distribution. From an indistinguishability adversary $\mathcal{A}^{\mathsf{ind-atk}}$ with advantage ϵ, we will construct an anonymity adversary $\mathcal{A}^{\mathsf{ano-atk}}$ as follows.

Let $(\mathsf{pk}_0, \mathsf{pk}_1)$ be the input to $\mathcal{A}^{\mathsf{ano-atk}}$. $\mathcal{A}^{\mathsf{ano-atk}}$ will run $\mathcal{A}^{\mathsf{ind-atk}}$ on pk_0. Queries made by $\mathcal{A}^{\mathsf{ind-atk}}$ will be simply passed to $\mathcal{A}^{\mathsf{ano-atk}}$'s own challenger. Note that pk_1 is independent of the view of $\mathcal{A}^{\mathsf{ind-atk}}$.

At some time, $\mathcal{A}^{\mathsf{ano-atk}}$ gets from his challenger a challenge (c, k) and is asked to tell the key (pk_0 or pk_1) under which it was created. $\mathcal{A}^{\mathsf{ano-atk}}$ will forward this challenge to $\mathcal{A}^{\mathsf{ind-atk}}$. In case it was created under pk_1, since pk_1 (together with the corresponding private key) is sampled from $\mathsf{PK} \times \mathsf{SK}$, Property C implies that $k = \mathcal{K}.\mathsf{decap}_{\mathsf{sk}_1}(c)$ is a uniformly random element of K. Therefore, the value k is either the decapsulation of c under pk_0, or a uniformly random element in K, and thus compatible with the game $\mathcal{A}^{\mathsf{ind-atk}}$ is designed to play.

Further queries by $\mathcal{A}^{\mathsf{ind-atk}}$ continue to be handled as before. At the end, $\mathcal{A}^{\mathsf{ind-atk}}$ will output a bit representing his guess for k being the decapsulation of c under the public key pk_0 or not. $\mathcal{A}^{\mathsf{ano-atk}}$ will use this bit as his guess for the key under which k was created. It is clear that:

$$\mathsf{Adv}(\mathcal{A}^{\mathsf{ano-atk}}) = \mathsf{Adv}(\mathcal{A}^{\mathsf{ind-atk}}).$$

Now assume that the distribution on K is *only indistinguishable* from uniform. Let $\mathcal{A}^{\mathsf{ind-atk}}$ be an indistinguishability distinguisher. If the advantage of $\mathcal{A}^{\mathsf{ind-atk}}$ in the reduction described above is non-negligibly different from the advantage of $\mathcal{A}^{\mathsf{ind-atk}}$ in a real attack, then $\mathcal{A}^{\mathsf{ind-atk}}$ can be easily used as a distinguisher for the distribution considered by Property C. As a consequence:

$$\mathsf{Adv}(\mathcal{A}^{\mathsf{ano-atk}}) \approx \mathsf{Adv}(\mathcal{A}^{\mathsf{ind-atk}}),$$

where $\approx$ means "equal up to negligible terms".

□

Theorem 2.4. *Let $\mathcal{K}$ be a key encapsulation mechanism. Given $(t, q) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$; if $\mathcal{K}$ is (t, ε, q)-IND-ATK secure, then it is $(t, \frac{\varepsilon}{2}, q)$-IND-ATK secure, where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{CCA}\}$.*

Proof. Given an anonymity adversary $\mathcal{A}^{\mathsf{ano-atk}}$ with advantage ϵ, we will create an indistinguishability adversary $\mathcal{A}^{\mathsf{ind-atk}}$ in the same attack model ATK. Let pk_0 be the input to $\mathcal{A}^{\mathsf{ind-atk}}$. $\mathcal{A}^{\mathsf{ind-atk}}$ will run keygen to generate a public key pk_1 together with its corresponding private key sk_1.

Queries made by $\mathcal{A}^{\mathsf{ano-atk}}$ are answered in the following way: if they are with respect to the key pk_0, they are forwarded to $\mathcal{A}^{\mathsf{ind-atk}}$'s own challenger. Otherwise, in case they are with respect to pk_1, they are answered by $\mathcal{A}^{\mathsf{ind-atk}}$ using the private key sk_1.

Eventually, $\mathcal{A}^{\mathsf{ind-atk}}$ receives (c, k) from his own challenger, where k is either the decapsulation of c with respect to the key pk_0 or a uniformly chosen element from K. $\mathcal{A}^{\mathsf{ind-atk}}$ will forward his challenge to $\mathcal{A}^{\mathsf{ano-atk}}$.

Queries by $\mathcal{A}^{\mathsf{ano-atk}}$ continue to be handled as before.

If k corresponds to the decapsulation of c (under pk_0), then with overwhelming probability it is not the decapsulation of c under pk_1 (pk_1 (along with sk_1) was produced by $\mathcal{A}^{\mathsf{ind-atk}}$ and therefore it is independent of the view of his challenger who generates the challenge (c, k)). Otherwise, it is a random element in K, and with overwhelming probability it is not the decapsulation of c under either key.

At the end of the game, $\mathcal{A}^{\mathsf{ano-atk}}$ outputs a guess b' on the key used to decapsulate c in k. $\mathcal{A}^{\mathsf{ind-atk}}$ will then output the same guess b' to his challenger.

We have $\mathsf{Adv}(\mathcal{A}^{\mathsf{ano-atk}}) = \left|\epsilon = \Pr(b' = 0|b = 0) - \frac{1}{2}\right|$. In fact, $\mathcal{A}^{\mathsf{ano-atk}}$ is expected to work only when k is the decapsulation of c under pk_0 (corresponds to $b = 0$), which explains the conditional probability. In this case, $\mathcal{A}^{\mathsf{ano-atk}}$ is considered successful when he recognizes that k is decapsulation of c under pk_0.

The advantage of $\mathcal{A}^{\mathsf{ind-atk}}$ is by definition $\left|\Pr(b' = b) - \frac{1}{2}\right|$ and we have:

$$\begin{aligned}
\mathsf{Adv}(\mathcal{A}^{\mathsf{ind-atk}}) &= \left|\Pr(b' = b) - \frac{1}{2}\right| = \left|\Pr(b' = 0, b = 0) + \Pr(b' = 1, b = 1) - \frac{1}{2}\right| \\
&= \left|\Pr(b' = 0|b = 0)\Pr(b = 0) + \Pr(b' = 1|b = 1)\Pr(b = 1) - \frac{1}{2}\right| \\
&\approx \left|(\epsilon + \frac{1}{2})\frac{1}{2} + \frac{1}{2}\frac{1}{2} - \frac{1}{2}\right| = \frac{\epsilon}{2}
\end{aligned}$$

The last inequality, due to $\Pr(b' = 1|b = 1) \approx \frac{1}{2}$, is explained by the fact that in case $b = 1$, there is a negligible chance for k to be the decapsulation of c under pk_1. □

2.4.2 Data encapsulation mechanisms (DEMs)

DEMs are secret key encryption algorithms. They are, similarly to public key encryption, given by the same three algorithms (keygen, encrypt and decrypt), with the exception of generating only one key in the keygen algorithm which will serve for encryption as well as for decryption.

The security notion for DEMs, that corresponds to the ANO-CPA notion for public key encryption, is the *anonymity under a one time attack*; we define it as follows

Definition 2.7 (Anonymity (DEMs) - ANO-OT)**.** *Let $\mathcal{D} = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ be a DEM, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, for $b \xleftarrow{R} \{0,1\}$, where κ is a security parameter:*

Experiment $\mathbf{Exp}^{\textit{ano-ot}-b}_{\mathcal{D},\mathcal{A}}(\kappa)$
$k_0 \leftarrow \mathcal{D}.\mathsf{keygen}(\kappa)$,
$k_1 \leftarrow \mathcal{D}.\mathsf{keygen}(\kappa)$,
$(m^\star, \mathcal{I}) \leftarrow \mathcal{A}(\textit{find})$
$e^\star \leftarrow \Gamma.\mathsf{encrypt}_{k_b}(m^\star)$
$d \leftarrow \mathcal{A}(\textit{guess}, \mathcal{I}, e^\star)$
Return d

We define the advantage *of $\mathcal{A}$* via*:*

$$\mathbf{Adv}^{\mathsf{ano-ot}}_{\mathcal{D},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ano-ot-b}}_{\mathcal{D},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $t \in \mathbb{N}$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a (t,ε)-ANO-OT adversary against $\mathcal{D}$ if, running in time t, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{ano-ot}}_{\mathcal{D},\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme $\mathcal{D}$ is said to be (t,ε)-ANO-OT secure if no (t,ε)-ANO-OT adversary against it exists. Finally, we consider a DEM $\mathcal{D}$ with security parameter $\kappa \in \mathbb{N}$; $\mathcal{D}(\kappa)$ is said to be ANO-OT secure if, for any polynomial function $t : \mathbb{N} \to \mathbb{N}$ and any non-negligible function $\varepsilon : \mathbb{N} \to [0,1]$, it is $(t(\kappa), \varepsilon(\kappa))$-ANO-OT secure.

Note that the above notion corresponds to the ANO-CPA notion in the public key world because the adversary does not have any oracle access. In fact, in the secret key scenario, the adversary cannot even encrypt messages of his choice (chosen plaintext attack) since he does not have the key at his disposal.

It is easy to see that the same analysis, provided in Section 2.2, of the relation between anonymity and indistinguishability for public key encryption applies also here for DEMs. Moreover, it can be shown that one can obtain an ANO-CPA-secure encryption scheme from an ANO-CPA-secure KEM combined with an ANO-OT-secure DEM. The proof is similar to that of the indistinguishability notion, which is given in [Herranz *et al.*, 2006]. Finally, we introduce the following security notion for DEMs which captures both the indistinguishability and the anonymity under a one-time attack[2]:

Definition 2.8 (Invisibility (DEMs) - INV-OT)**.** *Let $\mathcal{D} = (\mathsf{keygen}, \mathsf{encrypt}, \mathsf{decrypt})$ be a DEM with ciphertext space C, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, for $b \xleftarrow{R} \{0,1\}$, where κ is a security parameter:*

[2]Later in this thesis, we will show how to use INV-OT and ANO-OT secure DEMs, combined with secure KEMs and secure digital signatures in order to build efficient and secure opaque signatures.

Experiment $\mathbf{Exp}^{\textit{inv-ot}-b}_{\mathcal{D},\mathcal{A}}(\kappa)$

$k \leftarrow \mathcal{D}.\mathsf{keygen}(\kappa)$,
$(m^\star, \mathcal{I}) \leftarrow \mathcal{A}(\textit{find}, k)$
$e^\star \leftarrow \Gamma.\mathsf{encrypt}_k(m^\star)$
if $b = 0$ then $\{e \xleftarrow{R} \mathsf{C}, e^\star \leftarrow e\}$
$d \leftarrow \mathcal{A}(\textit{guess}, \mathcal{I}, e^\star)$
Return d

We define the advantage *of* $\mathcal{A}$*,* via*:*

$$\mathbf{Adv}^{\mathsf{inv-ot}}_{\mathcal{D},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{inv-ot-b}}_{\mathcal{D},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $t \in \mathbb{N}$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* (t,ε)*-INV-OT adversary against* $\mathcal{D}$ *if, running in time* t*,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{inv-ot}}_{\mathcal{D},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* $\mathcal{D}$ *is said to be* (t,ε)*-INV-OT secure if no* (t,ε)*INV-OT adversary against it exists. Finally, we consider a DEM* $\mathcal{D}$ *with security parameter* κ*;* $\mathcal{D}(\kappa)$ *is said to be INV-OT secure if, for any any polynomial function* $t : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa))$*-INV-OT secure.*

2.5 Conclusion

In this chapter, we proved that key privacy and data privacy in encryption schemes are related to a certain extent. In fact, under some conditions, we showed that one notion yields the other. This allows to use existing work on the data privacy of some schemes in order to derive their anonymity. Moreover, we defined the anonymity notion for key and data encapsulation mechanisms and provided a study on the equivalence between this notion and the indistinguishability notion in these mechanisms.

Part II

Generic Constructions of Confirmer Signatures

Chapter 3

Overview of Confirmer Signatures

Abstract. Designated Confirmer signatures were introduced to limit the verification property inherent to digital signatures. In fact, the verification in these signatures is replaced by a confirmation/denial protocol between the *designated confirmer* and the signature recipient.
In this chapter, we give a short overview of designated confirmer signatures; we will start with the motivation behind such signatures, then provide the formal definition of these signatures as well as of their security properties, and finally, we will browse through the different realizations of these signatures from basic cryptographic primitives.

3.1 Motivation and definition

Digital signatures capture most of the properties met by signatures in the paper world, for instance the universal verification. However, in some applications, this property is not desired or at least needs to be controlled. The typical applications where we wish to restrain the holder of a signature from convincing other parties of the validity of the signature in question are:

Licensing software [Chaum & van Antwerpen, 1990] A software vendor is willing to embed signatures in his products such that only the paying customers are entitled to check the authenticity of these products. Moreover, he does not wish these paying customers to convince other parties of the genuineness of his goods.

Contract signing [Goldwasser & Waisbard, 2004] An employer issues a job offer to a certain candidate. Naturally, the employer needs to compete with the other job offers in order to attract the good candidate. Therefore, he does not wish the offer to be revealed to his competitors. At the same time, the candidate needs more than a verbal or unsigned agreement in order to protect himself from the employer not keeping his promise. Finally, when the candidate

accepts the offer, the employer wishes to *convert* the job offer he has issued to a publicly verifiable one, instead of having to issue a new contact.

Undeniable signatures were introduced in [Chaum & van Antwerpen, 1990] for this purpose; they proved critical in situations where privacy or anonymity is a big concern, e.g. licensing software [Chaum & van Antwerpen, 1990], electronic cash [Chaum & Pedersen, 1993; Boyd & Foo, 1998; Pointcheval, 2001] and electronic voting and auctions. In these signatures, the verification can be only attained by means of a cooperation with the signer, called the confirmation/denial protocols. Unfortunately, this very virtue (verification with only the signer's help) became their major shortcoming for many practical applications. The flaw was later repaired in [Chaum, 1995] by introducing the concept of *designated confirmer signatures*. In fact, this concept involves three entities, namely the signer who produces the signature, the designated confirmer who confirms or denies the alleged signature, and finally the recipient of the signature. Actually, in the literature, there is a clear separation between designated confirmer signatures or confirmer signatures for brevity, and *directed signatures* [Lim & Lee, 1993] which share the same concept as confirmer signatures with the exception of allowing both the signer and the confirmer to confirm/deny signatures. Finally, a desirable property in confirmer signatures is the convertibility of the signatures to ordinary ones. Indeed, such a property turned out to play a central role in fair payment protocols [Boyd & Foo, 1998].

Syntax

A convertible designated confirmer signature (CDCS) scheme consists of the following procedures:

Key generation (keygen**).** This algorithm inputs a security parameter κ and generates probabilistically two key pairs $(\mathsf{sk}_S, \mathsf{pk}_S)$ and $(\mathsf{sk}_C, \mathsf{pk}_C)$ for the signer and for the confirmer respectively.

ConfirmedSign (confirmedSign**).** On input sk_S, pk_C, and a message m, the signer outputs a confirmer signature signature μ, then interacts with the signature recipient (via an interactive protocol) to convince him of the validity of the just generated signature.

Verification (verify**).** This is an algorithm, run by the signer on a *just generated* signature or by the confirmer on *any* signature, to verify the validity of the alleged signature. The input to the algorithm is, in addition to the public keys pk_S and pk_C, the message, and the alleged signature, the random nonces r_S used to produce the signature in case the algorithm is run by the signer, or the private key sk_C in case the algorithm is run by the confirmer. The output of this algorithm is either 1 if the purported signature if valid on the message, or 0 otherwise.

Confirmation/denial protocols (confirm/deny**).** These are interactive protocols between the confirmer and a signature recipient (the verifier). Their common input consists of, in addition to pk_S and pk_C, the alleged signature μ, and the message m in question. The confirmer uses

his private key sk_C to convince the verifier of the validity (invalidity) of the signature μ on m. At the end, the verifier either accepts or rejects the proof.

Selective conversion (convert**).** This is an algorithm run by the confirmer, on a message m and its corresponding signature μ, using sk_C, in addition to pk_C and pk_S. The result is either $\perp$ in case μ is invalid w.r.t m, or a string which can be universally verified as a valid digital signature on the message m w.r.t. pk_S.

Selective verification (verifyConverted**).** This is an algorithm for verifying converted signatures. It inputs the converted signature, the message, pk_S, and pk_C, and outputs either 0 or 1.

Remark 3.1. *In [Gentry* et al.*, 2005; Wang* et al.*, 2007], the authors give the possibility of obtaining* directly *digital signatures on a given message. We find this unnecessary since it is already enough that a CDCS scheme supports the convertibility feature. Moreover, in [Wikström, 2007], the author considers a further protocol used by the confirmer to prove the correctness of the conversion. Throughout this thesis, we will mention the constructions that extend to this augmented model.*

Remark 3.2 (Security parameter)**.** *In the rest of this part, the security parameter of a construction consists of a tuple that comprises the security parameters used for the construction's building blocks. Thus, when we invoke the key generation or the setup algorithms of a construction's building block on input a given security parameter, say κ, we mean that we call the mentioned algorithms on input the field in κ which corresponds to the security parameter of the building block in question. The same remark applies for security; when we say that a construction's component is secure for the security parameter κ, we mean that it is secure w.r.t. the field in κ corresponding to the security parameter of this component.*

3.2 Security model

Since their introduction, many definitions and security models for CDCS have emerged. We present in this section the security properties we adhere to in this thesis. A security property is, as commonly agreed on, an attribute allowing a cryptographic scheme to withstand malicious attempts aiming at make it deviate from its prescribed task. These malicious attempts can be classified into two categories:

1. Attempts conducted by adversaries *inside* the system. This is for instance the case where the scheme operators are dishonest, coerced, or where they simply have their private keys compromised or stolen.

2. Attempts conducted by adversaries *outside* the system. These are the default attacks any cryptographic scheme should take into consideration.

A cryptographic scheme resilient against the first type of attacks is said to procure security in an *insider model*, whereas a scheme resilient against the second type of attacks is said to be secure in an *outsider model*. Consideration of the appropriate security model depends upon the functionality of the scheme; for some schemes it is enough to consider outsider security, for others it is imperative to consider insider security at least for some scheme properties.

The rest of this section will be devoted to the definition of the security properties we opt for, as well as to the comparison of these properties with the popular ones found in the literature.

Let CS be a CDCS scheme given by the algorithms/protocols keygen, confirmedSign, verify, confirm/deny, convert, and verifyConverted.
We assume that keygen inputs a security parameter κ and generates the key pairs $(\mathsf{sk}_S, \mathsf{pk}_S)$ and $(\mathsf{sk}_C, \mathsf{pk}_C)$ for the signer and for the confirmer respectively.
Let $\mathcal{M}$ and $\mathcal{S}$ be the message and signature spaces considered by CS respectively. Let further the confirmedSign (probabilistic) procedure produce a signature $\mu \in \mathcal{S}$ and a protocol (S, V) between the signer S and the verifier V (the signature recipient). Finally, we denote by r_S the randomness used in the confirmedSign procedure to generate the signature μ.

3.2.1 Completeness

The CDCS scheme CS is complete when it satisfies the following properties:

1. Every signature produced following the CS.confirmedSign procedure should be validated by the CS.verify algorithm. Moreover, if the signer and the signature holder are honest, then the signer must be able to confirm every valid signature he has just generated.

$$\begin{gathered}\forall m \in \mathcal{M}, \text{if } \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S, \mathsf{pk}_S, \mathsf{pk}_C\}}(m) = \{\mu, (S, V)\} \text{ then :} \\ \mathsf{CS.verify}_{\{\mathsf{pk}_S, \mathsf{pk}_C, r_S\}}(m, \mu) = 1, \text{ and} \\ \Pr[(S, V)(m, \mu, \mathsf{pk}_C, \mathsf{pk}_S) = \mathsf{Reject}] = \mathsf{negl}(\kappa),\end{gathered}$$

   where the probability is taken over the random tosses of both the prover and the verifier, and negl is a negligible function.

2. The conversion of every signature produced following the CS.confirmedSign procedure should be a string which can be universally verified as a valid digital signature on the message in question.

$$\begin{gathered}\forall m \in \mathcal{M}, \text{if } \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S, \mathsf{pk}_S, \mathsf{pk}_C\}}(m) = \{\mu, (S, V)\} \text{ then :} \\ \sigma = \mathsf{CS.convert}_{\mathsf{sk}_C}(m, \mu) \Rightarrow \mathsf{CS.verifyConverted}_{\{\mathsf{pk}_S, \mathsf{pk}_C\}}(m, \sigma) = 1.\end{gathered}$$

3. If the confirmer and the signature holder are honest, then the confirmer must be able to confirm every valid signature, i.e. every signature validated by the algorithm CS.verify, and disavow every invalid signature.

$$\forall m \in \mathcal{M},\ \forall \mu \in \mathcal{S}:$$
$$\mathsf{CS.verify}_{\{\mathsf{pk}_S,\mathsf{pk}_C,\mathsf{sk}_C\}}(m,\mu) = 1 \Rightarrow \Pr[\mathsf{CS.confirm}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S) = \mathsf{Reject}] = \mathsf{negl}(\kappa),$$
$$\mathsf{CS.verify}_{\{\mathsf{pk}_S,\mathsf{pk}_C,\mathsf{sk}_C\}}(m,\mu) = 0 \Rightarrow \Pr[\mathsf{CS.deny}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S) = \mathsf{Reject}] = \mathsf{negl}(\kappa),$$

where the probability is taken over the random tosses of both the prover and the verifier, and negl is a negligible function.

3.2.2 Security for the verifier

This property informally means that an adversary who compromises the private keys of both the signer and the confirmer cannot convince the verifier of the validity (invalidity) of an invalid (a valid) confirmer signature. That is, the protocols confirmedSign, confirmation and denial are *sound*. It is obvious that we consider security in the insider model for this property. In fact, we require the genuiness of the signatures despite their opacity. The formal definition of this property is as follows.

Definition 3.1. *Let $\mathcal{A}$ be an adversary against the confirmer signature scheme* CS. *We consider the following experiment:*

1. *$\mathcal{A}$ is given $(\mathsf{sk}_S, \mathsf{pk}_S)$ and $(\mathsf{sk}_C, \mathsf{pk}_C)$, output of the algorithm* CS.keygen.
2. *$\mathcal{A}$ produces a message m. He also runs* CS.confirmedSign *on m and produces a signature μ using $\mathsf{sk}_S, \mathsf{pk}_S$ and pk_C. Finally, $\mathcal{A}$ produces a string μ', from the confirmer signatures space, such that* $\mathsf{CS.verify}_{\{\mathsf{pk}_S,\mathsf{pk}_C,\mathsf{sk}_C\}}(m,\mu') = 0$.
3. *$\mathcal{A}$ interacts with a verifier V on the common input (m, μ') and executes the protocol $(\mathcal{A}, V)$, as a part of the* CS.confirmedSign *algorithm, in addition to the protocol* CS.confirm. *Moreover, $\mathcal{A}$ interacts with V on the common input (m, μ) and runs the protocol* CS.deny.

CS *is said to provide security for the verifier if the following equations hold:*

$$\Pr[(\mathcal{A},V)(m,\mu',\mathsf{pk}_C,\mathsf{pk}_S) = \mathsf{Accept}] = \mathsf{negl}(\kappa), \tag{3.1}$$
$$\Pr[\mathsf{CS.confirm}(m,\mu',\mathsf{pk}_C,\mathsf{pk}_S) = \mathsf{Accept}] = \mathsf{negl}(\kappa), \tag{3.2}$$
$$\Pr[\mathsf{CS.deny}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S) = \mathsf{Accept}] = \mathsf{negl}(\kappa), \tag{3.3}$$

where the probability is over all the random tosses of $\mathcal{A}$ and V, and negl *is a negligible function.*

3.2.3 Security for the signer

Security for the signer informally means that no one (including the confirmer) except the signer can issue valid confirmer signatures; it is then clear that this security property considers insider adversaries (the confirmer).

The formal definition of this requirement is as follows.

Definition 3.2 (Security for the signer)**.** *We consider the CDCS scheme* CS *described earlier in this section. Let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment:*

$\boxed{\textit{Experiment } \mathbf{Exp}^{\mathsf{euf-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa)}$
$(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{CS.keygen}(\kappa)$
$(\mathsf{pk}_C, \mathsf{sk}_C) \leftarrow \mathcal{A}(\mathsf{pk}_S)$

$(m^\star, \mu^\star) \leftarrow \mathcal{A}^{\mathfrak{S}}(\mathsf{pk}_S, \mathsf{pk}_C, \mathsf{sk}_C)$
$\mathfrak{S} : m \longmapsto \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S, \mathsf{pk}_S, \mathsf{pk}_C\}}(m)$
return 1 if and only if the following properties are satisfied:
- $\mathsf{CS.verify}_{\{\mathsf{pk}_S, \mathsf{pk}_C, \mathsf{sk}_C\}}[m^\star, \mu^\star] = 1$
- $m^\star$ *was not queried to* $\mathfrak{S}$

We define the success *of* $\mathcal{A}$ via*:*

$$\mathbf{Succ}^{\textit{euf-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = \Pr\left[\mathbf{Exp}^{\textit{euf-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = 1\right].$$

Given $(t, q_s) \in \mathbb{N}^2$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* (t, ε, q_s)*-EUF-CMA adversary against* CS *if, running in time* t *and issuing* q_s *queries to the* CS.confirmedSign *oracle,* $\mathcal{A}$ *has* $\mathbf{Succ}^{\textit{euf-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* CS *is said to be* (t, ε, q_s)*-EUF-CMA secure if no* (t, ε, q_s)*-EUF-CMA adversary against it exists. Finally, we consider a CDCS scheme* CS *with security parameter* $\kappa \in \mathbb{N}$*;* $\mathsf{CS}(\kappa)$ *is said to be EUF-CMA secure if, for any polynomial functions* $t,\ q_s : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_s(\kappa))$*-EUF-CMA secure.*

Remark 3.3. *Note that the adversary* $\mathcal{A}$ *in the above definition is not given the oracles* CS.verify*,* CS.confirm/CS.deny*, and* CS.convert*. In fact, these oracles are useless for him as he has the confirmer private key* sk_C *at his disposal.*

3.2.4 Security for the confirmer

This is the crucial property for confirmer signatures as it quantifies their opacity. We can divide it into two sub-properties: non-transferability which refers to the inability of the verifier to transfer his conviction about the validity/invalidity of a signature to a third party, and opacity which refers to the inability of a verifier to decide on the validity/invalidity of a signature w.r.t. a given message.

The first property can be ensured if the protocols CS.confirmedSign, CS.confirm and CS.deny are zero knowledge, that is if the transcript resulting from the interaction of the verifier with the signer or the confirmer during these protocols can be efficiently simulated.

The second property is a bit intricate. First, there is the question of whether to consider insider or only outsider adversaries. Insider security means that the signer's private key can be compromised in which case the entire system is broken. However, it (insider security) might be needed in situations where we want to protect the invisibility of signatures issued by the genuine signer from

an adversary who has stolen this signer's private key. The second issue concerning the opacity of the signatures is whether one should hide the validity of the signatures w.r.t. the message in question or hide the identity of the signer. In the rest of this subsection, we will describe formally the non-transferability of confirmer signatures as well as the different notions of their opacity.

Non-transferability

Let CS be the CDCS scheme described above. Non-transferability of CS.confirmedSign and of CS.confirm/CS.deny is defined through the following two games involving the adversary, the signer, the confirmer, and a simulator:

Game 1: the adversary is given the public keys of the signer and of the confirmer, namely pk_S and pk_C resp. He can then make arbitrary queries of type CS.confirmedSign to the signer and of type CS.confirm/CS.deny and CS.convert to the confirmer. Note that the adversary is allowed at any time to create his own key pairs $(\mathsf{sk}_{S'}, \mathsf{pk}_{S'})$ and query the confirmer for verification/conversion of signatures w.r.t. these key pairs. Eventually, the adversary presents two strings m and μ for which he wishes to carry out, on the common input $(m, \mu, \mathsf{pk}_S, \mathsf{pk}_C)$, the protocol CS.confirmedSign with the signer, or the protocols CS.confirm/CS.deny with the confirmer. The private input of the signer is the randomness used to generate the signature μ (in case μ is a signature just generated by the signer), whereas the private input of the confirmer is his private key sk_C. The adversary continues issuing queries to both the signer and the confirmer until he decides that this phase is over and produces an output.

Game 2: this game is similar to the previous one with the difference of playing a simulator instead of running the real signer or the real confirmer when it comes to the interaction of the adversary with the signer in CS.confirmedSign or with the confirmer in CS.confirm/CS.deny on the common input $(m, \mu, \mathsf{pk}_S, \mathsf{pk}_C)$. The simulator is not given the private input of neither the signer nor the confirmer. It is however allowed to issue a single oracle call that tells whether μ is a valid confirmer signature on m w.r.t. pk_S and pk_C. Note that the simulator in this game refers to a probabilistic polynomial Turing machine with rewind.

The signatures issued by CS are said to be non-transferable if there exists an efficient simulator such that for all $(\mathsf{pk}_S, \mathsf{pk}_C)$, the outputs of the adversary in Game 1 and Game 2 are indistinguishable.

Remark 3.4. *The notion of non-transferability is very close to the notion of zero knowledge in the sense that both notions assume the existence of an efficient algorithm (the simulator) capable of producing transcripts of the proof/protocol in question that are indistinguishable from those obtained from the interaction with the real prover. The only difference is that in non-transferability, we require that the simulator interacts with the adversary, whereas in zero knowledge transcripts are enough. However, according to Remark 1.9, the ZK property of the* CS.confirmedSign *or the* CS.confirm/CS.deny *protocols is enough to ensure the non-transferability of the confirmer signatures.*

Invisibility

Invisibility against a chosen message attack (INV-CMA) for the confirmer signature scheme CS is defined through the following game between an attacker $\mathcal{A}$ and his challenger $\mathcal{R}$:

Phase 1: after $\mathcal{A}$ gets the public parameters of CS, namely pk_S and pk_C, from $\mathcal{R}$, he starts issuing queries of type CS.confirmedSign, CS.confirm/CS.deny, and CS.convert in an adaptive way.

Challenge: once $\mathcal{A}$ decides that **Phase 1** is over, he outputs two messages $m_0^\star, m_1^\star$ and requests a challenge signature $\mu^\star$. $\mathcal{R}$ picks uniformly at random a bit $b \in \{0,1\}$. Then $\mu^\star$ is generated using the CS.confirmedSign algorithm on the message $m_b^\star$.

Phase 2: $\mathcal{A}$ resumes adaptively making the previous types of queries, with the exception of not querying $(m_i^\star, \mu^\star)$, $i = 0, 1$, to the CS.$\{$confirm, deny$\}$ and CS.convert oracles. At the end, $\mathcal{A}$ outputs a bit b'. He wins the game if $b = b'$.

Definition 3.3 (Invisibility (INV-CMA))**.** *Let* CS *be the CDCS scheme described earlier, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment for $b \xleftarrow{R} \{0,1\}$:*

Experiment $\mathbf{Exp}^{\textit{inv-cma}-b}_{\mathsf{CS},\mathcal{A}}(\kappa)$

$(\mathsf{pk}_S, \mathsf{sk}_S, \mathsf{pk}_C, \mathsf{sk}_C) \leftarrow \mathsf{CS.keygen}(\kappa)$

$(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{find}, \mathsf{pk}_S, \mathsf{pk}_C)$

$\mathfrak{S} : m \longmapsto \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m)$

$\mathfrak{Cv} : (m,\mu) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

$\mu^\star \leftarrow \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m_b^\star)$

$d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{guess}, \mathcal{I}, \mu^\star, \mathsf{pk}_S, \mathsf{pk}_C)$

$\mathfrak{S} : m \longmapsto \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m)$

$\mathfrak{Cv} : (m,\mu)(\neq (m_i^\star,\mu^\star), i = 0,1) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu)(\neq (m_i^\star,\mu^\star), i = 0,1) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{inv-cma-b}}_{\mathsf{CS},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a $(t, \varepsilon, q_s, q_v, q_{sc})$-INV-CMA adversary against CS *if, running in time t and issuing q_s queries to the* CS.confirmedSign *oracle, q_v queries to the* CS.confirm/CS.deny *oracles, and q_{sc} queries to the* CS.convert *oracle, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme* CS *is said to be $(t, \varepsilon, q_s, q_v, q_{sc})$-INV-CMA secure if no $(t, \varepsilon, q_s, q_v, q_{sc})$-INV-CMA adversary against it exists. Finally, we consider a CDCS scheme* CS *with security parameter*

$\kappa \in \mathbb{N}$; $\mathsf{CS}(\kappa)$ *is said to be INV-CMA secure if, for any polynomial functions* $t, q_s, q_v, q_{sc} : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_s(\kappa), q_v(\kappa), q_{sc}(\kappa))$*-INV-CMA secure.*

Anonymity

Anonymity against a chosen message attack (ANO-CMA) for the confirmer signature scheme CS is defined through the following game between an attacker $\mathcal{A}$ and his challenger $\mathcal{R}$.

Phase 1: $\mathcal{A}$ gets two public keys for CS, namely two key pairs $(\mathsf{sk}_{S_0}, \mathsf{pk}_{S_0})$ and $(\mathsf{sk}_{S_1}, \mathsf{pk}_{S_1})$ for the signer, and two key pairs $(\mathsf{sk}_{C_0}, \mathsf{pk}_{C_0})$ and $(\mathsf{sk}_{C_1}, \mathsf{pk}_{C_1})$ for the confirmer. He then issues queries to the $\mathsf{CS}.\mathsf{confirmedSign}$, $\mathsf{CS}.\mathsf{confirm}/\mathsf{CS}.\mathsf{deny}$, and $\mathsf{CS}.\mathsf{convert}$ oracles, w.r.t. both keys, in an adaptive way.

Challenge: once $\mathcal{A}$ decides that *Phase 1* is over, he outputs a messages $m^\star$ and requests a challenge signature $\mu^\star$. $\mathcal{R}$ picks uniformly at random a bit $b \in \{0,1\}$, then $\mu^\star$ is generated using the $\mathsf{CS}.\mathsf{confirmedSign}$ algorithm on the message $m^\star$ w.r.t. $(\mathsf{pk}_{S_b}, \mathsf{pk}_{C_b})$.

Phase 2: $\mathcal{A}$ resumes adaptively making the previous types of queries, with the exception of not querying $(m^\star, \mu^\star)$ to the $\mathsf{CS}.\{\mathsf{confirm}, \mathsf{deny}\}$ and $\mathsf{CS}.\mathsf{convert}$ oracles of both keys $(\mathsf{pk}_{S_b}, \mathsf{pk}_{C_b})$, $b = 0, 1$. At the end, $\mathcal{A}$ outputs a bit b'. He wins the game if $b = b'$.

Definition 3.4 (Anonymity (ANO-CMA))**.** *Let* CS *be the CDCS scheme defined earlier, and let* $\mathcal{A}$ *be a PPTM. We consider the following random experiment for* $b \xleftarrow{R} \{0,1\}$*:*

Experiment $\mathbf{Exp}^{\textit{ano-cma}-b}_{\mathsf{CS},\mathcal{A}}(\kappa)$

$(\mathsf{pk}_{S_0}, \mathsf{sk}_{S_0}, \mathsf{pk}_{C_0}, \mathsf{sk}_{C_0}) \leftarrow \mathsf{CS}.\mathsf{keygen}(\kappa)$

$(\mathsf{pk}_{S_1}, \mathsf{sk}_{S_1}, \mathsf{pk}_{C_1}, \mathsf{sk}_{C_1}) \leftarrow \mathsf{CS}.\mathsf{keygen}(\kappa)$

$(m^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{find}, \mathsf{pk}_{S_0}, \mathsf{pk}_{S_1}, \mathsf{pk}_{C_0}, \mathsf{pk}_{C_1})$

$\mathfrak{S} = \mathfrak{S}_i, i = 0,1; \mathfrak{S}_i : m \longmapsto \mathsf{CS}.\mathsf{confirmedSign}_{\{\mathsf{sk}_{S_i}, \mathsf{pk}_{S_i}, \mathsf{pk}_{C_i}\}}(m).$

$\mathfrak{Cv} = \mathfrak{Cv}_i, i = 0,1; \mathfrak{Cv}_i : (m,\mu) \longmapsto \mathsf{CS}.\mathsf{convert}_{\mathsf{sk}_{C_i}}(m,\mu).$

$\mathfrak{V} = \mathfrak{V}_i, i = 0,1; \mathfrak{V}_i : (m,\mu) \longmapsto \mathsf{CS}.\{\mathsf{confirm}, \mathsf{deny}\}(m, \mu, \mathsf{pk}_{C_i}, \mathsf{pk}_{S_i}).$

$\mu^\star \leftarrow \mathsf{CS}.\mathsf{confirmedSign}_{\{\mathsf{sk}_{S_b}, \mathsf{pk}_{S_b}, \mathsf{pk}_{C_b}\}}(m^\star)$

$d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{guess}, \mathcal{I}, \mu^\star)$

$\mathfrak{S} = \mathfrak{S}_i, i = 0,1; \mathfrak{S}_i : m \longmapsto \mathsf{CS}.\mathsf{confirmedSign}_{\{\mathsf{sk}_{S_i}, \mathsf{pk}_{S_i}, \mathsf{pk}_{C_i}\}}(m).$

$\mathfrak{Cv} = \mathfrak{Cv}_i, i = 0,1; \mathfrak{Cv}_i : (m,\mu)(\neq (m^\star, \mu^\star)) \longmapsto \mathsf{CS}.\mathsf{convert}_{\mathsf{sk}_{C_i}}(m,\mu).$

$\mathfrak{V} = \mathfrak{V}_i, i = 0,1; \mathfrak{V}_i : (m,\mu)(\neq (m^\star, \mu^\star)) \longmapsto \mathsf{CS}.\{\mathsf{confirm}, \mathsf{deny}\}(m, \mu, \mathsf{pk}_{S_i}, \mathsf{pk}_{C_i}).$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{ano-cma-b}}_{\mathsf{CS},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a $(t, \varepsilon, q_s, q_v, q_{sc})$-ANO-CMA adversary against CS *if, running in time t and issuing q_s queries to the* CS.confirmedSign *oracle, q_v queries to the* CS.confirm/CS.deny *oracles, and q_{sc} queries to the* CS.convert *oracle, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{ano-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme* CS *is said to be $(t, \varepsilon, q_s, q_v, q_{sc})$-ANO-CMA secure if no $(t, \varepsilon, q_s, q_v, q_{sc})$-ANO-CMA adversary against it exists. Finally, we consider a CDCS scheme* CS *with security parameter $\kappa \in \mathbb{N}$; $\mathsf{CS}(\kappa)$ is said to be ANO-CMA secure if, for any polynomial functions $t, q_s, q_v, q_{sc} : \mathbb{N} \to \mathbb{N}$ and any non-negligible function $\varepsilon : \mathbb{N} \to [0,1]$, it is $(t(\kappa), \varepsilon(\kappa), q_s(\kappa), q_v(\kappa), q_{sc}(\kappa))$-ANO-CMA secure.*

Strong invisibility

To capture both anonymity and invisibility, Galbraith and Mao introduced in [Galbraith & Mao, 2003] a notion, which we denote SINV-CMA, that requires the confirmer signatures to be indistinguishable from random elements in the signature space. This new notion is proven to imply both INV-CMA and ANO-CMA (Theorem 1 and Theorem 4 respectively of [Galbraith & Mao, 2003]). This notion is defined exactly as the INV-CMA notion with the difference that when it comes to the challenge phase, the adversary produces a message m and the challenge signature is either a valid confirmer signature on m, issued according to confirmedSign, or a random string from the confirmer signatures space.

Definition 3.5 (Strong Invisibility (SINV-CMA))**.** *Let* CS *be the CDCS scheme, described earlier, with confirmer signatures space* S*, and let $\mathcal{A}$ be a PPTM. We consider the following random experiment for $b \xleftarrow{R} \{0,1\}$:*

Experiment $\mathbf{Exp}^{\mathsf{sinv\text{-}cma-}b}_{\mathsf{CS},\mathcal{A}}(\kappa)$

$(\mathsf{pk}_S, \mathsf{sk}_S, \mathsf{pk}_C, \mathsf{sk}_C) \leftarrow \mathsf{CS.keygen}(\kappa)$

$(m^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{find}, \mathsf{pk}_S, \mathsf{pk}_C)$

$\mathfrak{S} : m \longmapsto \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m)$

$\mathfrak{Cv} : (m,\mu) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

$\mu^\star \leftarrow \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m^\star)$

if $b = 0$ then $\{\mu \xleftarrow{R} \mathsf{S}, \mu^\star \leftarrow \mu\}$

$d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{Cv},\mathfrak{V}}(\textit{guess}, \mathcal{I}, \mu^\star, \mathsf{pk}_S, \mathsf{pk}_C)$

$\mathfrak{S} : m \longmapsto \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m)$

$\mathfrak{Cv} : (m,\mu)(\neq (m^\star,\mu^\star)) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu)(\neq (m^\star,\mu^\star)) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{sinv-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{sinv-cma-b}}_{\mathsf{CS},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* $(t, \varepsilon, q_s, q_v, q_{sc})$*-SINV-CMA adversary against* CS *if, running in time* t *and issuing* q_s *queries to the* CS.confirmedSign *oracle,* q_v *queries to the* CS.confirm/CS.deny *oracles, and* q_{sc} *queries to the* CS.convert *oracle,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{sinv-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* CS *is said to be* $(t, \varepsilon, q_s, q_v, q_{sc})$*-SINV-CMA secure if no* $(t, \varepsilon, q_s, q_v, q_{sc})$*-SINV-CMA adversary against it exists. Finally, we consider a CDCS scheme* CS *with security parameter* $\kappa \in \mathbb{N}$*;* $\mathsf{CS}(\kappa)$ *is said to be SINV-CMA secure if, for any polynomial functions* $t, q_s, q_v, q_{sc} : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_s(\kappa), q_v(\kappa), q_{sc}(\kappa))$*-SINV-CMA secure.*

3.2.5 Comparison with other security models

In this paragraph, we compare our security model with the popular ones found in the literature:

- Our definitions of completeness, security for the verifier and non-transferability of the confirmedSign, confirmation, and denial protocols are the same provided as in [Camenisch & Michels, 2000; Gentry *et al.*, 2005; Wang *et al.*, 2007].

- We consider the *insider security model against malicious confirmers* in our definition for unforgeability. I.e. the adversary is *allowed* to choose his key pair $(\mathsf{sk}_C, \mathsf{pk}_C)$. This is justified by the need of preventing the confirmer from impersonating the signer by issuing valid signatures on his behalf. Hence, our definition of unforgeability, which is the same as the one considered by [Wikström, 2007], implies its similars in [Camenisch & Michels, 2000; Gentry *et al.*, 2005; Wang *et al.*, 2007].

- Our definition of invisibility (INV-CMA), oppositely to the definitions in [Camenisch & Michels, 2000; Gentry *et al.*, 2005; Wang *et al.*, 2007], is considered in the *outsider security model*. I.e. the adversary does not know the private key of the signer. We justify this by considering the CDCS scheme broken if the signer is corrupted or coerced. Actually, "outsider security might be all one needs" for invisibility as phrased by the authors in [An *et al.*, 2002].

- Our definition of invisibility (INV-CMA), oppositely to the definitions in [Gentry *et al.*, 2005; Wang *et al.*, 2007], allows the signer to sign the same message many times without loss of invisibility, which is needed in licensing software.

- Finally our definition of invisibility (INV-CMA), like the definitions in [Camenisch & Michels, 2000; Gentry *et al.*, 2005; Wang *et al.*, 2007] and unlike the definition in [Galbraith & Mao,

2003], does not guarantee the non-transferability of the signatures. I.e. the confirmer signature might convince the recipient that the signer was involved in the signature of some message. We refer to the discussion in [Gentry *et al.*, 2005] (Section 3) for techniques that can be used by the signer to camouflage the presence of valid signatures, e.g. the signer can for instance publish a few "dummy" signatures during each time period.

3.3 Constructions

Since the introduction of confirmer signatures, a number of attempts have been made to produce them from basic primitives. The first construction is due to Okamoto [Okamoto, 1994], and was used to prove equivalence between confirmer signatures and public key encryption with respect to existence. Thus, efficiency was not taken into account in the framework. The subsequent proposals follow one of the following two strategies; either produce a digital signature on the message to be signed, then encrypt the resulting signature, or produce a commitment on the message, encrypt the string used to generate the commitment, and finally sign the latter. We recall in this section the constructions realizing those two approaches along with their security analyses.

3.3.1 The "encryption of a signature" paradigm

This approach consists in first producing a digital signature on the message to be signed, then encrypting the produced signature using a suitable public key encryption scheme. The construction was first formally[1] described in [Camenisch & Michels, 2000], and required the components to meet the highest security notions (EUF-CMA signatures and IND-CCA encryption). The main weakness of the construction lies in the resort to zero knowledge (ZK) protocols of general NP statements in the confirmation/denial protocol.

The construction

Let Σ be a digital signature scheme given by $\Sigma.\mathsf{keygen}$ which generates a key pair (private key = $\Sigma.\mathsf{sk}$, public key= $\Sigma.\mathsf{pk}$), $\Sigma.\mathsf{sign}$, and $\Sigma.\mathsf{verify}$. Let furthermore Γ denote a public key encryption scheme described by $\Gamma.\mathsf{keygen}$ that generates the key pair (private key = $\Gamma.\mathsf{sk}$, public key= $\Gamma.\mathsf{pk}$), $\Gamma.\mathsf{encrypt}$, and $\Gamma.\mathsf{decrypt}$.

Finally, let $m \in \{0,1\}^\star$ be a message. The construction is as follows:

Setup (setup**).** On input the security parameter κ, output the public parameters of Γ and Σ.

Key generation (keygen**).** Invoke the algorithms $\Sigma.\mathsf{keygen}$ and $\Gamma.\mathsf{keygen}$ to generate the keys $\Sigma.\mathsf{sk}$, $\Sigma.\mathsf{pk}$, $\Gamma.\mathsf{sk}$, and $\Gamma.\mathsf{pk}$. Set the signer's key pair to $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$ and the confirmer's key pair to $(\Gamma.\mathsf{sk}, \Gamma.\mathsf{pk})$.

[1] The idea without proof was already known, for instance, it was mentioned in [Damgård & Pedersen, 1996].

ConfirmedSign (confirmedSign**).** On a message m, the signer first computes a (digital) signature $\sigma = \Sigma_{\Sigma.\mathsf{sk}}.\mathsf{sign}(m)$ on m, then encrypts the result using $\Gamma.\mathsf{encrypt}$. The resulting ciphertext $\mu = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(\sigma)$ forms the output confirmer signature. Moreover, the signer interacts with the signature recipient in a zero knowledge protocol where he (the signer) proves that the output is a valid confirmer signature on the message in question. The prover's private input is the randomness used to generate the encryption μ of σ.

Verification (verify**).** To check whether an alleged confirmer signature μ, issued on a certain message m, is valid, the confirmer first decrypts it to recover σ, then calls the algorithm $\Sigma.\mathsf{verify}$ on the result using $\Sigma.\mathsf{pk}$. The signature is valid if and only if the output of the latter item is 1. We stress again that this algorithm is run by the confirmer. It can also be run by the signer on a *just generated signature* μ; using the randomness used to generate μ (as encryption of some σ), the signer checks whether μ is well formed, i.e. whether μ is indeed an encryption of σ, then he checks, using $\Sigma.\mathsf{pk}$, whether σ is a valid digital signature on m.

Confirmation/Denial protocol (confirm/deny**).** To confirm (deny) a purported signature μ on a certain message m, the confirmer first checks its validity using the verification algorithm. According to the result, the signer issues a zero knowledge proof of knowledge of the decryption of μ, that passes (does not pass) $\Sigma.\mathsf{verify}$.

Selective conversion (convert**).** Given a signature μ on m, the confirmer first checks whether it is valid. If it is the case, then he outputs $\Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(\mu)$, otherwise he outputs $\perp$.

Selective verification (verifyConverted**).** It is easy to see that the verification of converted signatures can be achieved by the algorithm $\Sigma.\mathsf{verify}$ using $\Sigma.\mathsf{pk}$.

Remark 3.5. *It is possible to issue the confirmation/denial protocols as well as the one underlying the* confirmedSign *algorithm because the underlying assertions define either NP (in case of* confirmedSign *or* confirm*) or co-NP (in case of* deny*) languages which accept zero knowledge proof systems according to Subsection 1.4.2.*

Security analysis

The completeness of the construction above is ensured by the correctness of the algorithms $\Sigma.\mathsf{sign}$, $\Sigma.\mathsf{verify}$, $\Gamma.\mathsf{encrypt}$ and $\Gamma.\mathsf{decrypt}$, and by the completeness of the proofs underlying the protocols confirmedSign, confirm and deny. As for the security for the verifier and the non-transferability of the signatures, they are established thanks to the soundness and zero knowledgeness of the proofs underlying the protocols confirmedSign, confirm, and deny. Moreover, the resulting signatures are existentially unforgeable against malicious confirmers, and they are invisible in the insider model.

Theorem 3.1. *Given* $(t, q_s) \in \mathbb{N}^2$ *and* $\varepsilon \in [0, 1]$*, the construction depicted above is* (t, ϵ, q_s)*-EUF-CMA secure* against malicious confirmers *if the underlying signature scheme is also* (t, ϵ, q_s)*-EUF-CMA secure.*

The proof is similar to that provided in [Camenisch & Michels, 2000] although the latter one does not explicitly prove the construction to be unforgeable against *malicious confirmers*.

Proof. Let $\mathcal{A}$ be a (t, ϵ, q_s)-EUF-CMA adversary against the construction. We will construct a (t, ϵ, q_s)-EUF-CMA adversary $\mathcal{R}$ against the underlying digital signature scheme as follows.

$\mathcal{R}$ gets the public key of the signature scheme Σ from his challenger. Then he chooses a suitable encryption scheme Γ and gets from $\mathcal{A}$ the generated confirmer key pair $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$.

Signature queries made by $\mathcal{A}$ on a message m_i can be answered as follows. First, $\mathcal{R}$ requests his challenger for a digital signature σ_i on m_i, then he encrypts σ_i in μ_i and outputs the result to $\mathcal{A}$. Finally, he interacts with $\mathcal{A}$ in a protocol where he proves that the generated signature is indeed a valid confirmer signature on m_i. The private input of $\mathcal{R}$ in this protocol is the randomness used to encrypt σ_i in μ_i, or $\Gamma.\mathsf{sk}$. Note that $\mathcal{A}$ can check the validity of this signature himself using $\Gamma.\mathsf{sk}$.

Eventually, $\mathcal{A}$ outputs a pair (m, μ) consisting of a message m that was never queried for signature and a valid confirmer signature μ on it. $\mathcal{R}$ will simply output $\sigma = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(\mu)$ to his own challenger. In fact, σ is a valid digital signature on the message m which was never queried by $\mathcal{R}$ to his own challenger, and thus forms a valid existential forgery on Σ. □

Theorem 3.2. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $\varepsilon \in [0, 1]$, the construction above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure* in the insider model *if the underlying encryption scheme is $(t, \epsilon, q_v + q_{sc})$-IND-CCA secure.*

We give a sketch of the proof below and we refer to the full version in [Camenisch & Michels, 2000].

Sketch. Let $\mathcal{R}$ be an IND-CCA adversary against an encryption scheme Γ. $\mathcal{R}$ gets the public key $\Gamma.\mathsf{pk}$ of the encryption scheme from his challenger and is further given an INV-CMA adversary $\mathcal{A}$ against the construction depicted above.

$\mathcal{R}$ will choose a digital signature scheme Σ along with a key pair $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$ and will provide $\mathcal{A}$ with the public parameters of the confirmer signature resulting from combining the encryption scheme Γ and the signature Σ. $\mathcal{A}$ will get also hold of the private signing key, namely $\Sigma.\mathsf{sk}$.

Simulation of the $\mathsf{confirmedSign}$ queries made by $\mathcal{A}$ is done as the ordinary algorithm would perform, namely by first producing a digital signature, using $\Sigma.\mathsf{sk}$, on the message in question, then encrypting the resulting signature using $\Gamma.\mathsf{pk}$. The resulting ciphertext forms the confirmer signature output to $\mathcal{A}$. $\mathcal{R}$ will then interact with $\mathcal{A}$ to prove the validity of the just generated signature. The private input of $\mathcal{R}$ in such a protocol is the randomness used to encrypt the digital signature.

Simulation of the $\mathsf{confirm}/\mathsf{deny}$ queries (m, μ) is done by first invoking the decryption oracle of Γ on μ to obtain $\sigma = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(\mu)$, then checking the validity of σ w.r.t. m. According to the result, $\mathcal{R}$ issues a simulation of the $\mathsf{confirm}/\mathsf{deny}$ protocols. In fact, since the confirmer signatures are non-transferable, then there exists a simulation of the confirmation/denial protocols which is indistinguishable from the real execution of these protocols. Simulation of the $\mathsf{convert}$ queries is done by simply decrypting (using the decryption oracle) the confirmer signature in question.

Eventually, $\mathcal{A}$ outputs two messages m_0, m_1 and requests a challenge confirmer signature. At that point, $\mathcal{R}$ produces a digital signature $\sigma_i = \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(m_i)$ on each message m_i, $i = 0, 1$. With overwhelming probability, $\sigma_0 \neq \sigma_1$ as $m_0 \neq m_1$. Then, $\mathcal{R}$ gives these two signatures to his own challenger. He gets as a challenge ciphertext $\mu^\star$, which is either the encryption of σ_0 or σ_i, that he will forward to $\mathcal{A}$.

$\mathcal{A}$ continues issuing confirmedSign, confirm/deny and convert queries as before . Note that at that point, and *according to the invisibility notion considered by the authors in [Camenisch & Michels, 2000]*, $\mathcal{A}$ is not allowed to issue confirm/deny and convert queries which involve $\mu^\star$. $\mathcal{R}$ can answer as previously, for instance he is able to invoke his decryption oracle without problems as the confirmer signatures in play are different from the challenge ciphertext $\mu^\star$.

Finally, when $\mathcal{A}$ outputs his guess (either 0 or 1) on the message underlying the signature $\mu^\star$, $\mathcal{A}$ will forward the same guess to his own challenger. □

Other variants

The Goldwasser-Waisbard [Goldwasser & Waisbard, 2004] construction. This construction was the first to circumvent, although partially, the main problem in the basic paradigm, namely the recourse to proofs of general NP statements in the confirmation/denial protocols. The idea consists in considering a class $\mathbb{S}$ of digital signatures which accept efficient *witness hiding proofs of knowledge (WHPOK)*. A WHPOK (see for instance [Goldreich, 2001, Section 4.6] for more details) is informally a proof where the prover does not reveal the witness but may leak some knowledge during his interaction with the verifier; it is then a weaker notion than zero knowledge. Let (t, b, s_b) be an accepting transcript resulting from the interaction, between a prover and a verifier, in which the prover convinces the verifier that he holds a digital signature σ on the common input message m. t forms the first message, or the commitment, sent by the prover. $b \xleftarrow{R} \{0, 1\}$ denotes the public coin, or the challenge sent by the verifier. Finally, s_b denotes the response of the prover to the challenge b. It is assumed that given two different accepting transcripts (t, b, s_b) and $(t, 1 - b, s_{1-b})$, there exists a knowledge extractor which can extract the witness, namely the signature σ. With such a class of signatures in addition to an IND-CCA secure encryption scheme Γ, the authors in [Goldwasser & Waisbard, 2004] provide confirmer signatures on a message m as follows:

1. The signer first produces a digital signature σ on m. Then, he computes the commitment t he would send to the verifier if he wishes to provide a WHPOK for σ. Next he computes s_0 and s_1, the responses to the challenges $b = 0$ and $b = 1$ resp., along with their encryptions e_0 and e_1 using random coins r_0 and r_1 resp. Finally the signer sends (t, e_0, e_1) to the signature recipient.
2. The signature recipient selects $b \xleftarrow{R} \{0, 1\}$ and sends it to the signer.
3. The signer reveals s_b to the verifier along with the random coin used to produce its encryption e_b, namely r_b.

4. The signature recipient accepts if e_b is indeed the encryption of s_b using r_b, and if (t, b, s_b) is indeed an accepting transcript for the WHPOK.

The triplet (t, e_0, e_1) forms the confirmer signature the verifier needs to present before the confirmer for verification or conversion. In fact, the confirmer can decrypt both e_0 and e_1 in s_0 and s_1 resp., then extract the witness σ in case of a valid signature and finally interact (in case of a confirmation query) with the verifier in a protocol similar to the one above. Conversion is done by revealing σ. And finally, the denial of an invalid signature consists of a ZK proof that the conversion returns an invalid signature.

The construction successfully gets rid of proofs of general NP statements in the confirmation protocol. However, it still resorts to them in the denial protocol. Moreover, the length of the signatures as well as their generation cost grow linearly with the number of rounds in the WHPOK. Finally, the security guarantees satisfied by the construction are much more relaxed compared to the ones met by the construction realizing the basic "encryption of a signature" paradigm. For instance, the non-transferability of the signatures may not be guaranteed with the use of WHPOK, as the adversary might get sufficient knowledge (from the confirmation protocol) to convince other parties with the validity of the signature he is holding. Also, the adversary is not given access to a conversion oracle in the non-transferability definition which means that one can say nothing about his ability in transferring knowledge of the validity of signatures when he sees some converted signatures.

The Wikström [Wikström, 2007] construction. This construction does not differ much from the basic "encryption of a signature" paradigm in that it consists in first producing a digital signature on the message to be signed then encrypting the resulting signature. The difference is that the used encryption scheme needs to support labels. Actually, the encryption of the digital signatures is done under the label $\Sigma.\mathsf{pk}$, which denotes the public key of the signer. An instantiation of the construction is further provided and is proved secure under the strong RSA assumption, the decision composite residuosity assumption, and the decision Diffie-Hellman problem. The basic novelty of the work [Wikström, 2007] lies in the new security model proposed for confirmer signatures, and in which the construction is analyzed. We summarize below the basic new security definitions proposed in [Wikström, 2007]:

1. *Security for the signer*. This property is a reformulation of the unforgeability property for confirmer signatures, which takes into condition malicious confirmers. I.e. the adversary is allowed to choose the confirmer key $(\mathsf{sk}_C, \mathsf{pk}_C)$. Almost all previous constructions , e.g. [Camenisch & Michels, 2000; Goldwasser & Waisbard, 2004; Gentry *et al.*, 2005; Wang *et al.*, 2007] extend to this model.
2. *Security for the confirmer*. This property, called in [Wikström, 2007] impersonation resistance, requires that no one should play the role of the genuine confirmer, namely prove that the confirmer key is well formed, that a signature is valid/invalid and finally that a conversion is correct. The formalization of such is property is done as usual

through a game where the adversary has access to a genuine confirmer oracle that he can consult up to the challenge phase. Consequently, one gets with this definition only a "lunch-time" security for the confirmer unlike the definitions proposed earlier in Subsection 3.2.4. The non-transferability of signatures proposed in [Wikström, 2007] is the same proposed earlier in this chapter.

The *Security for the verifier* property in [Wikström, 2007] is the same proposed in Subsection 3.2.2, which agrees with the definitions in [Camenisch & Michels, 2000; Goldwasser & Waisbard, 2004; Gentry *et al.*, 2005; Wang *et al.*, 2007]. Finally, it is worth mentioning that the model in [Wikström, 2007] requires the confirmer to prove the correctness of a conversion. Again, all the previous constructions,e.g. [Camenisch & Michels, 2000; Goldwasser & Waisbard, 2004; Gentry *et al.*, 2005; Wang *et al.*, 2007], as well as the ones we will encounter in this thesis extend to this model.

3.3.2 The "signature of a commitment" paradigm

This paradigm was first considered in [Michels & Stadler, 1998] to build confirmer signatures from signatures obtained using the Fiat-Shamir paradigm. The main criticism to such a construction lies in the resort to the ROM (resulting from the use of the Fiat-Shamir Paradigm) and the non-support of the convertibility feature. In [An *et al.*, 2002], the authors upgraded this technique to the "encrypt then commit then sign" method, which consists in first generating a random string, say r and encrypting it in e, then using r to generate a commitment c on the message to be signed, and finally produce a digital signature on the commitment c. This approach was used in the context of signcryption in [An *et al.*, 2002] and was analyzed in the insider security model. Later in [Gentry *et al.*, 2005], the authors used it to build confirmer signatures and provided an efficient instantiation using Camenisch-Shoup [Camenisch & Shoup, 2003]'s encryption and and Pedersen's commitment. The resulting construction was shown to be invisible in the insider security model if the underlying commitment is hiding and the underlying encryption is IND-CCA secure. However, the authors in [Wang *et al.*, 2007] disproved this claim by exhibiting an attack against the invisibility of the construction regardless of the underlying encryption: given the challenge signature (e, c, σ) on the message m_b, where $b \in \{0, 1\}$ and m_0, m_1 are the challenge messages output by the invisibility adversary $\mathcal{A}$, the latter computes a commitment c' such that the underlying message m' is meaningfully related to m_0, m_1 ($m' = k + m_b - m_0$, where k is known to $\mathcal{A}$) and the underlying random string is the same used to create c. Such a construction is possible using Pedersen's commitment. Next, $\mathcal{A}$ produces a digital signature σ' on c' (this is possible in the insider security model) and queries the conversion oracle on (e, c', σ') and the message k; if the oracle answers $r \neq \perp$, then $\mathcal{A}$ outputs $b = 0$, otherwise if the oracle answers $\perp$, $\mathcal{A}$ outputs $b = 1$. The authors in [Wang *et al.*, 2007] proposed a fix to this construction which consists in using *encryption schemes with labels*.

In the rest of this section, we describe the construction of [Wang *et al.*, 2007] and we recall its security analysis.

The construction in [Wang *et al.*, 2007]

Setup (setup**).** Consider a digital signature scheme Σ, an encryption scheme Γ with labels, and a commitment scheme Ω.

Key generation (keygen**).** The signer key pair consists of $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$, corresponding to the key pair of the signature scheme Σ, whereas the confirmer key pair consists of $(\Gamma.\mathsf{sk}, \Gamma.\mathsf{sk})$ which corresponds to the key pair related to Γ.

ConfirmedSign (confirmedSign**).** To sign a message m, the signer first computes a commitment c on the message, then encrypts in e, under the label $m\|\Sigma.\mathsf{pk}$, the random string used for the commitment, say r, and finally, signs the commitment c using $\Sigma.\mathsf{sk}$. The confirmer signature consists of the triple $(e, c, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c))$. Next, the signer interacts with the verifier in a protocol where he (the signer) proves in ZK the knowledge of r such that $r = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}, m\|\Sigma.\mathsf{pk}}(e)$ and $c = \Omega.\mathsf{commit}(m, r)$. Such a proof is possible to issue using the randomness used to encrypt r in e. In fact, the encryption and commitment algorithms in an encryption scheme and a commitment scheme resp. define an NP language that accepts a zero knowledge proof system.

Confirmation/Denial protocol (confirm/deny**).** To confirm/deny a signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a given message m, the confirmer first checks whether μ_3 is a valid digital signature on μ_2 w.r.t. $\Sigma.\mathsf{pk}$, if so, he provides a ZK proof (using his private key $\Gamma.\mathsf{sk}$) of the equality/inequality of the decryption of μ_1 (w.r.t. the label $m\|\Sigma.\mathsf{pk}$) and the opening value of the commitment μ_2 w.r.t. m. Again this proof is plausible since every NP (co-NP in case of inequality) language accepts a zero knowledge proof system.

Verification (verify**).** The verification of a purported signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a given message m is achieved by first checking the validity of μ_3 w.r.t. to m as a digital signature, then checking the equality of the decryption of μ_1 (w.r.t. the label $m\|\Sigma.\mathsf{pk}$) and the opening value of μ_3, as a commitment on m. This equality check can be achieved by the signer, who has just generated μ, given the randomness used to create the ciphertext μ_1, or by the confirmer who can decrypt μ_1 using $\Gamma.\mathsf{sk}$.

Selective conversion (convert**).** Selective conversion of a signature $\mu = (\mu_1, \mu_2, \mu_3)$ is achieved by releasing the decryption of μ_1, in case μ is valid, or the symbol $\perp$ otherwise.

Selective verification (verifyConverted**).** It is easy to see that the verification of converted signatures can be achieved by the algorithms $\Omega.\mathsf{open}$ and $\Sigma.\mathsf{verify}$.

Security analysis

Completeness, soundness and non-transferability of the confirmedSign and the confirmation/denial protocols follow directly from using zero knowledge proofs of knowledge. Concerning unforge-

ability of the resulting confirmer signatures, it rests on the EUF-CMA security and on the binding property of the underlying digital signature scheme and the commitment scheme respectively. Finally, invisibility is attained by using an IND-CCA secure encryption scheme with labels and a secure commitment scheme. Details about the proofs were not given so far, but are due to appear in a forthcoming paper (full version of [Wang *et al.*, 2007]). Since the paper is not available yet, we flesh out what we suspect to be the proofs in this paragraph.

Theorem 3.3. *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the construction depicted above is (t, ϵ, q_s)-EUF-CMA secure if uses a binding commitment scheme and a (t, ϵ, q_s)-EUF-CMA secure digital signature scheme.*

Proof. Let $\mathcal{A}$ be an attacker against the construction. We will construct an attacker $\mathcal{R}$ against the underlying signature scheme as follows.

$\mathcal{R}$ gets the parameters of the signature scheme Σ from his challenger, namely the public key $\Sigma.\mathsf{pk}$. Then, $\mathcal{R}$ will choose an appropriate encryption scheme Γ with labels and a commitment scheme Ω. $\mathcal{R}$ gets from $\mathcal{A}$ the generated confirmer key pair $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$ and finally sets the mentioned entities as components of the construction $\mathcal{A}$ is trying to attack.

For a signature query on a message m_i, $\mathcal{R}$ will first create a commitment c_i using a random string r_i, then he will query his own challenger for a digital signature on c_i. Let σ_i be the output digital signature on c_i. The output confirmer signature consists of the triple $\mu_i = (e_i, c_i, \sigma_i)$, where e_i is an encryption of r_i under the label $m_i \| \Sigma.\mathsf{pk}$.

$\mathcal{A}$ will have at his disposal $\Gamma.\mathsf{sk}$ and thus he won't need to ask confirm/deny or selective conversion queries. And, even in case he requests them, $\mathcal{R}$ is able to answer such queries with the knowledge of $\Gamma.\mathsf{sk}$.

At some point, $\mathcal{A}$ will output a forgery $\mu^\star = (e^\star, c^\star, \sigma^\star)$ on some message $m^\star$ that has never been queried. If there exists an $1 \leq i \leq q_s$ such that $c^\star = c_i$, where $\mu_i = (e_i, c_i, \sigma_i)$ is an output confirmer signature on a query m_i, then since $m_i \neq m^\star$, $\mathcal{R}$ will output a collision for the commitment scheme Ω. As the latter is by assumption binding, $c^\star$ never occurred in signatures output to $\mathcal{A}$. Therefore $(c^\star, \sigma^\star)$ corresponds to a valid existential forgery on Σ. □

The invisibility of the construction is considered in [Wang *et al.*, 2007] in a slightly different model and it rests on the security of the underlying encryption and commitment schemes. The main difference between the model in [Wang *et al.*, 2007] and our definition of invisibility, provided earlier, lies in giving the adversary the signer's private key, however disallowing him to make verification/conversion queries w.r.t. the challenge message and valid signatures on it.

Definition 3.6 (Invisibility [Wang *et al.*, 2007] (INV2-CMA))**.** *Let $\mathsf{CS} = (\mathsf{keygen}, \mathsf{confirmedSign}, \mathsf{verify}, \mathsf{confirm}/\mathsf{deny}, \mathsf{convert}, \mathsf{verifyConverted})$ be a CDCS scheme, and let $\mathcal{A}$ be a PPTM. We define the relation R between two strings μ and μ' w.r.t. a message m to be 1 if both μ and μ' are valid confirmer signatures on m (w.r.t. the same signer's key) and we write $R(m, \mu, \mu') = 1$. We consider the following random experiment, where κ is a security parameter, and $b \xleftarrow{R} \{0, 1\}$:*

Experiment $\mathbf{Exp}^{\textit{inv2-cma}-b}_{\mathsf{CS},\mathcal{A}}(\kappa)$

$(\mathsf{pk}_S, \mathsf{sk}_S, \mathsf{pk}_C, \mathsf{sk}_C) \leftarrow \mathsf{CS.keygen}(\kappa)$

$(m_0^\star, m_1^\star, \mathcal{I}) \leftarrow \mathcal{A}^{\mathfrak{Cv},\mathfrak{V}}(\textit{find}, \mathsf{pk}_S, \mathsf{sk}_S, \mathsf{pk}_C)$

$\mathfrak{Cv} : (m,\mu) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

$\mu^\star \leftarrow \mathsf{CS.confirmedSign}_{\{\mathsf{sk}_S,\mathsf{pk}_S,\mathsf{pk}_C\}}(m_b^\star)$

$d \leftarrow \mathcal{A}^{\mathfrak{Cv},\mathfrak{V}}(\textit{guess}, \mathcal{I}, \mu^\star, \mathsf{pk}_S, \mathsf{sk}_S, \mathsf{pk}_C)$

$\mathfrak{Cv} : (m,\mu)(\neq (m_i^\star,\tilde{\mu}) : R(m_i,\mu^\star,\tilde{\mu}) = 1, i = 0,1) \longmapsto \mathsf{CS.convert}_{\mathsf{sk}_C}(m,\mu)$

$\mathfrak{V} : (m,\mu)(\neq (m_i^\star,\tilde{\mu}) : R(m_i,\mu^\star,\tilde{\mu}) = 1, i = 0,1) \longmapsto \mathsf{CS.}\{\mathsf{confirm},\mathsf{deny}\}(m,\mu,\mathsf{pk}_C,\mathsf{pk}_S)$

Return d

We define the advantage *of* $\mathcal{A}$ via*:*

$$\mathbf{Adv}^{\mathsf{inv2-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) = \left| \Pr\left[\mathbf{Exp}^{\mathsf{inv2-cma-b}}_{\mathsf{CS},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q_v, q_{sc}) \in \mathbb{N}^3$ *and* $\varepsilon \in [0,1]$*,* $\mathcal{A}$ *is called a* $(t, \varepsilon, q_v, q_{sc})$*-INV2-CMA adversary against* CS *if, running in time* t *and issuing* q_v *queries to the* CS.confirm/CS.deny *oracles and* q_{sc} *queries to the* CS.convert *oracle,* $\mathcal{A}$ *has* $\mathbf{Adv}^{\mathsf{inv2-cma}}_{\mathsf{CS},\mathcal{A}}(\kappa) \geq \varepsilon$*. The scheme* CS *is said to be* $(t, \varepsilon, q_v, q_{sc})$*-INV2-CMA secure if no* $(t, \varepsilon, q_v, q_{sc})$*-INV2-CMA adversary against it exists. Finally, we consider a CDCS scheme* CS *with security parameter* $\kappa \in \mathbb{N}$*;* $\mathsf{CS}(\kappa)$ *is said to be INV2-CMA secure if, for any polynomial functions* $t, q_v, q_{sc} : \mathbb{N} \to \mathbb{N}$ *and any non-negligible function* $\varepsilon : \mathbb{N} \to [0,1]$*, it is* $(t(\kappa), \varepsilon(\kappa), q_v(\kappa), q_{sc}(\kappa))$*-INV2-CMA secure.*

Remark 3.6. *Note that the adversary in the above definition does not need a* CS.confirmedSign *oracle since he has the signing private key* sk_S*.*

We present in the sequel the invisibility analysis in the model considered by the authors in [Wang *et al.*, 2007].

Theorem 3.4. *Given* $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ *and* $\varepsilon \in [0,1]$*, the construction depicted earlier in this subsection is* $(t, \epsilon, q_v, q_{sc})$*-INV2-CMA if it uses an injective, binding, and* (t, ϵ_h)*-hiding commitment, and a* $(t, \frac{\epsilon+\epsilon_h}{2}, q_v + q_{sc})$*-IND-CCA secure encryption with labels.*

Before proving this theorem, we need the following lemma:

Lemma 3.5. *Let* Ω *and* Γ *be a commitment and a public key encryption schemes respectively. We consider the following game between an adversary* $\mathcal{A}$ *and his challenger* $\mathcal{R}$*:*

1. $\mathcal{R}$ *invokes the algorithms* $\Gamma.\mathsf{keygen}(\kappa)$ *to generate* $(\mathsf{pk}, \mathsf{sk})$*, where* κ *is a security parameter.*
2. $\mathcal{A}$ *outputs two messages* m_0 *and* m_1 *such that* $m_0 \neq m_1$ *to his challenger.*
3. *R generates two nonces* r_0 *and* r_1 *such that* $r_0 \neq r_1$*. Next, he chooses two bits* $b, b' \overset{R}{\leftarrow} \{0,1\}$ *uniformly at random. Finally, he outputs to* $\mathcal{A}$ $c_b = \Omega.\mathsf{commit}(m_b, r_{1-b'})$ *and* $e_{b'} = \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(r_{b'})$*.*

4. *$\mathcal{A}$ outputs a bit b_a representing his guess of c_b not being the commitment of m_b using the nonce $\Gamma.\mathsf{decrypt}(e_{b'})$. $\mathcal{A}$ wins the game if $b_a \neq b$, and we define his advantage to be*

$$\mathsf{Adv}(\mathcal{A}) = \left| \Pr[b \neq b_a] - \frac{1}{2} \right|,$$

where the probability is taken over the random tosses of both $\mathcal{A}$ and $\mathcal{R}$.

If Ω is injective, binding, and (t_h, ϵ_h)-hiding, then $\mathsf{Adv}(\mathcal{A})$ in the above game is equal to ϵ_h.

Proof. Let ϵ be the advantage of $\mathcal{A}$ in game above. We will construct an adversary $\mathcal{R}$ which breaks the hiding property of the used commitment with advantage ϵ.

- $\mathcal{R}$ gets from $\mathcal{A}$ the message m_0, m_1, and forwards them to his own challenger.
- $\mathcal{R}$ receives from his challenger the commitment $c_b = \Omega.\mathsf{commit}(m_b, r)$ for some $b \xleftarrow{R} \{0,1\}$ and some nonce r.
- $\mathcal{R}$ generates a nonce r' and outputs to $\mathcal{A}$ c_b and $e = \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(r')$.
- When $\mathcal{A}$ outputs a bit b_a, $\mathcal{R}$ outputs to his challenger $1 - b_a$.

If $\mathcal{A}$ can by some means get hold of r', then he can compute $c_i = \Omega.\mathsf{commit}(m_i, r')$, $i = 0, 1$. Since Ω is injective and binding then $c_b \neq \Omega.\mathsf{commit}(m_b, r')$ and $c_b \neq \Omega.\mathsf{commit}(m_{1-b}, r')$ respectively, i.e. $c_b \notin \{c_0, c_1\}$.

We have by definition:

$$\begin{aligned} \epsilon_h = \mathsf{Adv}(\mathcal{R}) &= \left| \Pr[1 - b_a = b] - \frac{1}{2} \right| \\ &= \left| \Pr[b_a \neq b] - \frac{1}{2} \right| \\ &= \mathsf{Adv}(\mathcal{A}) \end{aligned}$$

□

Remark 3.7. *Note that the above lemma holds true regardless of the used encryption Γ. For instance, it can be used with encryption schemes which support labels and which do not require any kind of security.*

Let us now prove Theorem 3.4.

Proof. We assume the existence of a $(t, \epsilon, q_v, q_{sc})$ invisibility adversary $\mathcal{A}$ against the construction, where the underlying commitment is injective, binding, and (t, ϵ_h)-hiding. We will construct a reduction $\mathcal{R}$ which $(t, \frac{\epsilon+\epsilon_h}{2}, q_v + q_{sc})$-IND-CCA breaks the underlying encryption scheme.

[Parameter generation] $\mathcal{R}$ gets the parameters of the encryption scheme Γ from his challenger. Then he will choose a signature scheme Σ (along with a key pair $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$) and a secure commitment scheme Ω. $\mathcal{R}$ will set the above entities as components of the construction $\mathcal{A}$ is trying to attack.

[$\mathsf{confirm}/\mathsf{deny}$ **and** $\mathsf{convert}$ **queries]** To confirm/deny an alleged signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ on a message m_i, $\mathcal{R}$ will proceed as follows. First he checks the validity of the digital signature μ_i^3 on μ_i^2, in case it is invalid, he will output $\perp$, otherwise he will obtain the decryption of μ_i^1 (from the decryption oracle thanks to the CCA attack model), r_i; if r_i is (is not) the same string used to compute the commitment μ_i^2, $\mathcal{R}$ will issue a zero knowledge proof of the equality (inequality) of the decryption of μ_i^1 and the string used for the commitment μ_i^2. $\mathcal{R}$ can issue these proofs without the knowledge of $\Gamma.\mathsf{sk}$ using the rewinding technique which consists in rewinding the verifier (the adversary $\mathcal{A}$) until his output agrees with what the simulator ($\mathcal{R}$) has generated (the proofs are ZK and thus simulatable, see Remark 1.9). Selective conversion is similarly carried out with the exception of issuing the decryption of μ_i^1 in case the confirmer signature is valid and $\perp$ otherwise.

[Challenge phase] At some point, $\mathcal{A}$ will output two messages m_0, m_1. $\mathcal{R}$ will then choose uniformly at random a bit $b \xleftarrow{R} \{0,1\}$, and generate two different nonces r_0 and r_1. $\mathcal{R}$ will output to his challenger the label $m_b \| \Sigma.\mathsf{pk}$ and the strings r_0, r_1. He receives then a ciphertext $e_{b'}$, encryption of $r_{b'}$, for some $b' \xleftarrow{R} \{0,1\}$. To answer his challenger, $\mathcal{R}$ will compute a commitment c_b on the message m_b using the string $r_{b''}$ where $b'' \xleftarrow{R} \{0,1\}$. Then, $\mathcal{R}$ will output $\mu = (e_{b'}, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ as a challenge signature to $\mathcal{A}$.

In case $e_{b'}$ is an encryption of $r_{b''}$ (that is if $b' = b''$), then μ corresponds to a valid confirmer signature on m_b. Otherwise, it is not a valid signature on neither m_b nor m_{1-b}. In fact, Ω is injective and c_b is a commitment on m_b using a string different from the decryption of $e_{b'}$ under the label $m_b \| \Sigma.\mathsf{pk}$. If the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of $\mathcal{A}$ in the attack described in Definition 3.6, then and according to Lemma 3.5, $\mathcal{A}$ can be easily used to break the hiding property of the underlying commitment.

[Post challenge Phase] $\mathcal{R}$ will continue to handle $\mathcal{A}$'s queries as before. Note that in this phase, $\mathcal{R}$ cannot query his challenger for the decryption of $e_{b'}$ under the label $m_b \| \Sigma.\mathsf{pk}$. $\mathcal{R}$ needs such a decryption query if $\mathcal{A}$ requests the verification (conversion) of $(e_{b'}, c, \sigma)$ on the message m_b, where σ is a valid digital signature on c, and c is a valid commitment on m_b using either r_0 or r_1. If such a query occurs, $\mathcal{R}$ will issue the denial protocol (output $\perp$). This differs from the real algorithm when $(e_{b'}, c, \sigma)$ is a valid confirmer signature on m_b; two cases manifest: either $c = c_b$ in which case such a signature is not allowed for verification/conversion according to Definition 3.6, or $c = \Omega.\mathsf{commit}(m_b, r_{1-b''})$ which is very unlikely to occur since $r_{1-b''}$ is external to $\mathcal{A}$.

[Final output] Let b_a be the bit output by $\mathcal{A}$. $\mathcal{R}$ will output b'' to his challenger in case $b = b_a$ and $1 - b''$ otherwise.

The advantage of $\mathcal{A}$ in such an attack is defined by

$$\begin{aligned}\epsilon &= \mathsf{Adv}(\mathcal{A}) = \left|\Pr[b_a = b|b' = b''] - \frac{1}{2}\right| \\ &= \max(\Pr[b_a = b|b' = b''] - \frac{1}{2}, \Pr[b_a \neq b|b' = b''] - \frac{1}{2})\end{aligned}$$

Moreover, and according to Lemma 3.5, we have in case $b' \neq b$":

$$\begin{aligned}\epsilon_h &= \left|\Pr[b_a \neq b|b' \neq b''] - \frac{1}{2}\right| \\ &= \max(\Pr[b_a \neq b|b' \neq b''] - \frac{1}{2}, \Pr[b_a = b|b' \neq b''] - \frac{1}{2})\end{aligned}$$

Let us assume without loss of generality that $\epsilon = \Pr[b_a = b|b' = b''] - \frac{1}{2}$ and $e_h = \Pr[b_a \neq b|b' \neq b''] - \frac{1}{2}$. The advantage of $\mathcal{R}$ is the given by:

$$\begin{aligned}\mathsf{Adv}(\mathcal{R}) &= \Pr[b = b_a, b' = b''] + \Pr[b \neq b_a, b' \neq b''] - \frac{1}{2} \\ &= \Pr[b = b_a|b' = b'']\Pr[b' = b''] + \Pr[b \neq b_a|b' \neq b'']\Pr[b' \neq b''] - \frac{1}{2} \\ &= \frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}(\frac{1}{2} + \epsilon_h) - \frac{1}{2} \\ &= \frac{\epsilon + \epsilon_h}{2}\end{aligned}$$

The last but one equation is due to the fact $\Pr[b' \neq b''] = \Pr[b' = b''] = \frac{1}{2}$ as $b'' \xleftarrow{R} \{0, 1\}$.

□

3.4 Conclusion

In this section, we presented the two basic approaches adhered to when building convertible confirmer signatures from basic primitives. The invisibility of both constructions was investigated in the *insider* security model, which requires the underlying encryption scheme to meet the highest

level of security, namely IND-CCA security. This impacts negatively the efficiency of the confirmation/denial protocols as they resort to proofs of general NP statements, e.g. proving knowledge of the plaintext underlying an IND-CCA encryption. Since insider security might be too strong than what is actually needed in most real life applications, it would be interesting to examine the invisibility of these constructions in the *outsider* security model with the hope of weakening the security assumptions on the underlying building blocks and consequently improving the efficiency of the construction in general, and of its confirmation/denial protocols in particular.

Chapter 4

The "Encryption of a Signature" Paradigm

Abstract. The "encryption of a signature" paradigm is the most intuitive way to obtain designated confirmer signatures; it consists in first generating a digital signature on the message to be signed, then encrypting the result using a suitable encryption scheme. This approach requires the constituents (encryption and signature schemes) to meet the highest security notions in order to achieve secure constructions in the insider security model.
In this chapter, we revisit this method and establish the necessary/minimal and sufficient assumptions on the building blocks in order to attain secure confirmer signatures in the outsider model. Our study concludes that the paradigm, used in its basic form, cannot allow a class of encryption schemes which is vital for the efficiency of the confirmation/denial protocols. Next, we propose a slight variation of the paradigm and we demonstrate its efficiency by explicitly describing its confirmation/denial protocols when instantiated with building blocks from a large class of signature/encryption schemes. Interestingly, the class of signatures we consider is very popular and has been for instance used to build efficient designated verifier signatures.
Parts of the results described in this chapter were published in [El Aimani, 2008] and [El Aimani, 2009b] at IndoCrypt 2008 and IndoCrypt 2009 resp.

4.1 Analysis of the plain paradigm

We consider the construction of the plain "encryption of a signature" paradigm depicted in Subsection 3.3.1. More precisely, let Σ be a digital signature scheme given by $\Sigma.\mathsf{keygen}$ which generates a key pair (private key = $\Sigma.\mathsf{sk}$, public key= $\Sigma.\mathsf{pk}$), $\Sigma.\mathsf{sign}$, and $\Sigma.\mathsf{verify}$. Let furthermore Γ denote an encryption scheme described by $\Gamma.\mathsf{keygen}$ that generates the key pair (private key = $\Gamma.\mathsf{sk}$, public key= $\Gamma.\mathsf{pk}$), $\Gamma.\mathsf{encrypt}$ and $\Gamma.\mathsf{decrypt}$. The construction is as follows:

Setup (setup**).** On input the security parameter κ, output the public parameters of Γ and Σ.

Key generation (keygen**).** Invoke the algorithms $\Sigma.\mathsf{keygen}$ and $\Gamma.\mathsf{keygen}$ to generate the keys $\Sigma.\mathsf{sk}$, $\Sigma.\mathsf{pk}$, $\Gamma.\mathsf{sk}$ and $\Gamma.\mathsf{pk}$. Set the signer's key pair to $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$ and the confirmer's key pair to $(\Gamma.\mathsf{sk}, \Gamma.\mathsf{pk})$.

ConfirmedSign (confirmedSign**).** Let m be the message to be signed. The signer first computes a (digital) signature $\sigma = \Sigma_{\Sigma.\mathsf{sk}}.\mathsf{sign}(m)$ on m, then encrypts it using $\Gamma.\mathsf{encrypt}$. The resulting ciphertext $\mu = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(\sigma)$ forms the output confirmer signature. Moreover, the signer interacts with the signature recipient in a zero knowledge protocol where he (the signer) proves that the output is a valid confirmer signature on the message in question. The prover's private input is the randomness used to generate the encryption μ of σ.

Verification (verify**).** To check whether an alleged confirmer signature μ, issued on a certain message m, is valid, the confirmer first decrypts it in σ, then calls the algorithm $\Sigma.\mathsf{verify}$ on the result using $\Sigma.\mathsf{pk}$. The signature is valid if and only if the output of the latter item is 1. We stress again that this algorithm is run by the confirmer. It can also be run by the signer on a *just generated signature* μ; using the randomness used to generate μ (as encryption of some σ), the signer checks whether μ is well formed, i.e.whether μ is indeed an encryption of σ, then he checks, using $\Sigma.\mathsf{pk}$, whether σ is a valid digital signature on m.

Confirmation/Denial protocol (confirm/deny**).** To confirm (deny) a purported signature μ on a certain message m, the confirmer first checks its validity using the verification algorithm. According to the result, the signer issues a zero knowledge proof of knowledge of the decryption of μ, that passes (does not pass) $\Sigma.\mathsf{verify}$.

Selective conversion (convert**).** Given a signature μ on m, the confirmer first checks whether it is valid. If it is the case, then he outputs $\Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(\mu)$, otherwise he outputs $\perp$.

Selective verification (verifyConverted**).** It is easy to see that the verification of converted signatures can be achieved by the algorithm $\Sigma.\mathsf{verify}$ using $\Sigma.\mathsf{pk}$.

In this section, we prove that the condition on the underlying signature scheme (EUF-CMA secure) is also necessary to achieve EUF-CMA secure confirmer signatures. Furthermore, we prove that IND-PCA secure encryption schemes are already enough, though a minimal requirement, to achieve INV-CMA signatures.

4.1.1 The exact unforgeability of the construction

Theorem 4.1. *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the above generic construction is (t, ϵ, q_s)-EUF-CMA secure if and only if the underlying digital signature scheme is (t, ϵ, q_s)-EUF-CMA secure.*

Proof. The If direction has been already proven (see Subsection 3.3.1). We prove now the other direction. Let $(m^\star, \sigma^\star)$ be an existential forgery against the digital signature scheme. One can derive a forgery against the confirmer signature by simply encrypting the signature $\sigma^\star$ using the public

key of the confirmer. Simulation of the attacker's environment is easy; the reduction $\mathcal{R}$ (EUF-CMA attacker against the confirmer signature) will forward the appropriate parameters (those concerning the underlying digital signature) to the EUF-CMA attacker $\mathcal{A}$ against the underlying signature scheme. For a signature query on a message m, $\mathcal{R}$ will first request his challenger for a confirmer signature μ that he decrypts using the private key of the confirmer ($\mathcal{R}$ has access to sk_C according to the EUF-CMA security game described in 3.2.3) in σ, which he ($\mathcal{R}$) will output to $\mathcal{A}$.

At the end, $\mathcal{A}$ outputs a valid digital signature $\sigma^\star$ on a message $m^\star$ that he has never queried for signature. $\mathcal{R}$ encrypts this signature in $\mu^\star$ using the public key of the confirmer and outputs the result as a valid existential forgery on $m^\star$ ($\mathcal{R}$ never queried $m^\star$ for a confirmer signature). □

4.1.2 The exact invisibility of the construction

In this paragraph, we prove that IND-PCA secure encryption schemes are a minimal and sufficient requirement to achieve INV-CMA secure confirmer signatures. To prove this assertion, we proceed as follows. We first show that the INV-CMA security of the resulting signatures cannot rest on the NM-CPA security of the underlying encryption scheme. We do this by means of an efficient *meta-reduction* that uses such a reduction (the algorithm reducing NM-CPA breaking the underlying encryption scheme to INV-CMA breaking the construction) to break the NM-CPA security of the encryption scheme. Thus, under the assumption that the encryption scheme is NM-CPA secure, the meta reduction forbids the existence of such a reduction. In case the encryption scheme is not NM-CPA secure, such a reduction will be useless. This result will rule out automatically all the other notions that are weaker than NM-CPA, namely OW-CPA and IND-CPA. Next, we use a similar technique to exclude the OW-CCA notion. The next security notion to be considered is IND-PCA. Luckily, this notion turns out to be sufficient to obtain INV-CMA secure signatures.

Note that meta-reductions have been successfully used in a number of important cryptographic results, e.g. the result in [Boneh & Venkatesan, 1998] which proves the impossibility of reducing factoring to the RSA problem, or the results in [Paillier & Vergnaud, 2005; Paillier, 2007] which show that some well known signatures which are proven secure in the random oracle cannot conserve the same security in the standard model. All those impossibility results are partial as they apply only for certain reductions. Our result is in a first stage also partial since it requires the reduction $\mathcal{R}$, trying to attack a certain property of an encryption scheme given by the public key $\Gamma.\mathsf{pk}$, to provide the adversary against the confirmer signature with the confirmer public key $\Gamma.\mathsf{pk}$. We will denote such reductions by *key-preserving* reductions, inheriting the name from a wide and popular class of reductions which supply the adversary with the same public key as its challenge. Such reductions were for instance used in [Paillier & Villar, 2006] to prove a separation between factoring and IND-CCA-breaking some factoring-based encryption schemes in the standard model. Our restriction to such a class of reductions is not unnatural since, to our best knowledge, all the reductions basing the security of the generic constructions of confirmer signatures on the security of their underlying components, feed the adversary with the public keys of these components (signature schemes, encryption schemes, and commitment schemes). Next, we use similar techniques

to [Paillier & Villar, 2006] to extend our impossibility results to arbitrary reductions.

Impossibility results for key-preserving reductions

Lemma 4.2. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV-CMA adversary $\mathcal{A}$ against the above construction to an NM-CPA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that NM-CPA breaks the encryption scheme in question.*

Let us first interpret this result. The lemma claims that under the assumption of the underlying encryption scheme being NM-CPA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces NM-CPA breaking the encryption scheme in question to INV-CMA breaking the construction, or if there exists such an algorithm, then the underlying encryption scheme is not NM-CPA secure, thus rendering such a reduction useless.

Proof. Let $\mathcal{R}$ be a key-preserving reduction that reduces NM-CPA breaking the encryption scheme underlying the construction to INV-CMA breaking the construction itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to NM-CPA break the same encryption scheme by simulating an execution of the INV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack. $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key, say $\Gamma.\mathsf{pk}$. $\mathcal{M}$, acting as the INV-CMA adversary $\mathcal{A}$ against the construction, queries $\mathcal{R}$ on $m_0, m_1 \xleftarrow{R} \{0,1\}^\star$ for confirmer signatures. Then, he queries the resulting strings μ_0, μ_1 (corresponding to the confirmer signatures on m_0 and m_1 respectively) for a selective conversion. Let σ_0 and σ_1 be the output (digital) signatures on m_0 and m_1 respectively. At that point, $\mathcal{M}$ inputs $\mathcal{D} = \{\sigma_0, \sigma_1\}$ to his own challenger as a distribution probability from which the plaintexts will be drawn. He gets in response a challenge encryption $\mu^\star$, of either σ_0 or σ_1 under $\Gamma.\mathsf{pk}$, and is asked to produce a ciphertext μ' whose corresponding plaintext is meaningfully related to the decryption of $\mu^\star$. To do this, $\mathcal{M}$ chooses uniformly at random a bit $b \xleftarrow{R} \{0,1\}$. Then, he queries the presumed confirmer signature $\mu^\star$ on m_b for a selective conversion. If the result is different from $\perp$, i.e. $\mu^\star$ is the encryption of σ_b, then $\mathcal{M}$ will output $\Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(\overline{\sigma_b})$ ($\overline{\sigma_b}$ refers to the bit-complement of the element σ_b) and the relation R: $R(m, m') = (m' = \overline{m})$. Otherwise, he will output $\Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(\overline{\sigma_{1-b}})$ and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating an INV-CMA attacker against the generic construction). □

Remark 4.1. *In the above proof, $\mathcal{R}$ may not behave as a standard INV-CMA challenger. For instance, he may produce inconsistent answers when $\mathcal{A}$ asks the signature of m_0 and m_1, the conversion of μ_0 and μ_1 w.r.t. m_0 and m_1 respectively, or the conversion of $\mu^\star$ w.r.t. m_0 or m_1. In this case, $\mathcal{M}$ cannot answer his NM-CPA challenge, however $\mathcal{A}$ is not either expected to answer his INV-CMA challenge, and therefore $\mathcal{R}$ will be compelled to solve his challenge without the help of $\mathcal{A}$; that is $\mathcal{R}$ will be useless as it is solving a challenge without the help of $\mathcal{A}$, i.e. an* easy *challenge.*

Actually, such an argument applies for all the impossibility results that will be used throughout this thesis; if the reduction provides an incorrect simulation causing the meta-reduction a failure in answering his challenge, then the adversary, played/simulated by this meta-reduction, is not neither expected to answer his challenge. In this case, the reduction will be useless as it is solving a challenge in polynomial time without the help of the adversary.

Lemma 4.3. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV-CMA adversary $\mathcal{A}$ against the above construction to a OW-CCA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the encryption scheme in question.*

As mentioned previously, this lemma claims that under the assumption of the underlying encryption scheme being OW-CCA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces OW-CCA breaking the encryption scheme in question to INV-CMA breaking the construction, or if there exists such an algorithm, then the underlying encryption scheme is not OW-CCA secure, thus rendering such a reduction useless.

Proof. The proof technique is similar to the one above. Let $\mathcal{R}$ be the key-preserving reduction that reduces OW-CCA breaking the encryption scheme underlying the construction to INV-CMA breaking the construction itself. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same encryption scheme by simulating an execution of the INV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack. $\mathcal{M}$ gets his challenge c and is equipped with a decryption oracle that he can query on all ciphertexts of his choice except of course on the challenge. $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key $\Gamma.\mathsf{pk}$ and the same challenge c. Obviously, all decryption queries made by $\mathcal{R}$, which are by definition different from the challenge ciphertext c, can be forwarded to $\mathcal{M}$'s own challenger. At some point, $\mathcal{M}$, acting as an INV-CMA attacker against the construction, will output two messages m_0, m_1 and gets as response a challenge signature $\mu^\star$ which he is required to tell to which message it corresponds. With overwhelming probability, $\mu^\star \neq c$, in fact, the challenge c is not the encryption of a certain σ such that σ is a valid digital signature on the message m_0 or the message m_1. Therefore, $\mathcal{M}$ queries his own challenger for the decryption of $\mu^\star$ (he can issue such a query since it is different from the challenge ciphertext). He checks whether the result, say σ, is a valid digital signature on m_0 or m_1. Then, he will simply output the result of this verification. Finally, when $\mathcal{R}$ outputs his answer, decryption of the ciphertext c, $\mathcal{M}$ will simply forward this result to his challenger. □

Remark 4.2. *In the above proof, if $\mathcal{R}$ gives c as a challenge confirmer signature to $\mathcal{A}$ (simulated by $\mathcal{M}$), then $\mathcal{A}$ cannot solve the INV-CMA challenge as $\mathcal{M}$ cannot invoke his decryption oracle on c. Since it is very unlikely that c corresponds to a valid confirmer signature on the challenge messages m_0 or m_1, then whatever is the answer of $\mathcal{A}$ (actually in this case, $\mathcal{A}$, simulated by $\mathcal{M}$ who launched $\mathcal{R}$ over c, can abort the invisibility game) to the challenge c, this answer will not help $\mathcal{R}$ solving his OW-CCA challenge since he already knows that c cannot be (with overwhelming*

probability) a valid confirmer signature on either messages m_0 or m_1. In other words, in this case, whatever $\mathcal{R}$ learns from $\mathcal{A}$, he can also learn it without $\mathcal{A}$, which corresponds to a reduction $\mathcal{R}$ solving a OW-CCA challenge in polynomial time without the help of $\mathcal{A}$, i.e. the reduction $\mathcal{R}$ is useless as it is solving an easy problem.

Remark 4.3. *Note that the success of the meta-reduction $\mathcal{M}$, in the above proofs, is identical to the success of the reduction $\mathcal{R}$. Moreover, the above results apply to any key-preserving reduction (from NM-CPA or OW-CCA breaking the encryption scheme to INV-CMA breaking the construction), for instance, they apply to the (key-preserving) reduction making the best possible use of INV-CMA adversaries against the construction.*

Theorem 4.4. *The encryption scheme underlying the above construction must be at least IND-PCA secure, in case the considered reduction is key-preserving, in order to achieve INV-CMA secure signatures.*

Proof. We proceed in this proof with elimination. Lemma 4.2 rules out the notion NM-CPA and thus the notions IND-CPA and OW-CPA. Moreover, Lemma 4.3 rules out OW-CCA and thus OW-PCA (and also OW-CPA). Thus, the next notion to be considered is IND-PCA. □

Remark 4.4. *The above theorem is only valid when the considered notions are those obtained from pairing a security goal GOAL $\in \{OW, IND, NM\}$ and an attack model ATK $\in \{CPA, PCA, CCA\}$. Presence of other notions will require an additional study. However, Lemmas 4.2 and 4.3 will be always of use when there exists a relation between these new notions and the notions OW-CCA and NM-CPA.*

Generalization to arbitrary reductions

To extend the results in the previous paragraph to arbitrary reductions, we first define the notion of *non-malleability of an encryption scheme key generator* through the following two games:
In **Game 0**, we consider an algorithm $\mathcal{R}$ trying to break an encryption scheme Γ , w.r.t. a public key $\Gamma.\mathsf{pk}$, in the sense of NM-CPA (or OW-CCA) using an adversary $\mathcal{A}$ which solves a problem A, perfectly reducible to OW-CPA breaking the encryption scheme Γ. In this game, $\mathcal{R}$ launches $\mathcal{A}$ over his own challenge key $\Gamma.\mathsf{pk}$ and some other parameters chosen freely by $\mathcal{R}$. We will denote by $\mathsf{Adv}_0(\mathcal{R}^{\mathcal{A}})$ the success probability of $\mathcal{R}$ in such a game, where the probability is taken over the random tapes of both $\mathcal{R}$ and $\mathcal{A}$. We further define $\mathsf{Succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) = \max_{\mathcal{R}} \mathsf{Adv}_0(\mathcal{R}^{\mathcal{A}})$ to be the success in **Game 0** of the best reduction $\mathcal{R}$ making the best possible use of the adversary $\mathcal{A}$. Note that the goal of **Game 0** is to include all key-preserving reductions $\mathcal{R}$ from NM-CPA (or OW-CCA) breaking the encryption scheme in question to solving a problem A, which is reducible to OW-CPA breaking the same encryption scheme.
In **Game 1**, we consider the same entities as in **Game 0**, with the exception of providing $\mathcal{R}$ with, in addition to $\mathcal{A}$, a OW-CPA oracle (i.e. a decryption oracle corresponding to Γ) that he can query w.r.t. any public key $\Gamma.\mathsf{pk}' \neq \Gamma.\mathsf{pk}$, where $\Gamma.\mathsf{pk}$ is the challenge public key of $\mathcal{R}$. Similarly, we

define $\mathsf{Adv}_1(\mathcal{R}^{\mathcal{A}})$ to be the success of $\mathcal{R}$ in such a game, and $\mathsf{Succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}) = \max_{\mathcal{R}} \mathsf{Adv}_0(\mathcal{R}^{\mathcal{A}})$ the success in **Game 1** of the reduction $\mathcal{R}$ making the best possible use of the adversary $\mathcal{A}$ and of the decryption (OW-CPA) oracle.

Definition 4.1. *An encryption scheme Γ is said to have a non-malleable key generator if $\Delta = max_{\mathcal{A}}|\mathsf{Succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}) - \mathsf{Succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A})|$ is negligible in the security parameter.*

This definition informally means that an encryption scheme has a non-malleable key generator if NM-CPA (or OW-CCA) breaking it w.r.t. a key pk is no easier when given access to a decryption (OW-CPA) oracle w.r.t. any public key $\mathsf{pk}' \neq \mathsf{pk}$.

We generalize now our impossibility results to arbitrary reductions as follows.

Theorem 4.5. *If the encryption scheme underlying the above construction has a non-malleable key generator, then it must be at least IND-PCA secure in order to achieve INV1-CMA secure confirmer signatures.*

To prove this theorem, we first need the following lemma (similar to Lemma 6 of [Paillier & Villar, 2006])

Lemma 4.6. *Let $\mathcal{A}$ be an adversary solving a problem A, reducible to OW-CPA breaking an encryption scheme Γ, and let $\mathcal{R}$ be an arbitrary reduction $\mathcal{R}$ that NM-CPA (OW-CCA) breaks an encryption scheme Γ given access to $\mathcal{A}$. We have*

$$\mathsf{Adv}(\mathcal{R}) \leq \mathsf{Succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}).$$

Proof. We will construct an algorithm $\mathcal{M}$ that plays **Game 1** with respect to a perfect oracle for $\mathcal{A}$ and succeeds in breaking the NM-CPA (OW-CCA) security of Γ with the same success probability of $\mathcal{R}$. Algorithm $\mathcal{M}$ gets a challenge w.r.t. a public key pk and launches $\mathcal{R}$ over the same challenge and the same public key. If $\mathcal{R}$ calls $\mathcal{A}$ on pk, then $\mathcal{M}$ will call his own oracle for $\mathcal{A}$. Otherwise, if $\mathcal{R}$ calls $\mathcal{A}$ on $\mathsf{pk}' \neq \mathsf{pk}$, $\mathcal{M}$ will invoke his own decryption oracle for pk' (OW-CPA oracle) to answer the queries. In fact, by assumption, the problem A is reducible to OW-CPA breaking Γ. Finally, when $\mathcal{R}$ outputs the result to $\mathcal{M}$, the latter will output the same result to his own challenger. □

The proof of Theorem 4.5 is similar to that of Theorem 5 in [Paillier & Villar, 2006]:

Proof. We first remark that the invisibility of the construction depicted above is perfectly reducible to OW-CPA breaking the encryption scheme underlying the construction. In fact, an invisibility adversary $\mathcal{A}$, given a challenge confirmer signature, can first decrypt it, then check, using the algorithm $\Sigma.\mathsf{verify}$ and $\Sigma.\mathsf{pk}$, whether the result is a valid digital signature on the message in question. Next, we note that the advantage of the meta-reduction $\mathcal{M}$ in the proof of Lemma 4.2 (Lemma 4.3) is the same as the advantage of any key-preserving reduction $\mathcal{R}$ reducing the invisibility of a given confirmer signature to the NM-CPA (OW-CCA) security of its underlying encryption scheme Γ.

For instance, this applies to the reduction making the best use of an invisibility adversary $\mathcal{A}$ against the construction. Therefore we have:

$$\mathsf{Succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) \leq \mathsf{Succ}(NM - CPA[\Gamma]),$$

where $\mathsf{Succ}(NM - CPA[\Gamma])$ is the success of breaking Γ in the NP-CPA sense. We also have

$$\mathsf{Succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) \leq \mathsf{Succ}(OW - CCA[\Gamma]).$$

Now, Let $\mathcal{R}$ be an arbitrary reduction from NM-CPA (OW-CCA) breaking an encryption scheme Γ, with a non-malleable key generator, to INV-CMA breaking the construction (using the same encryption scheme Γ). We have

$$\begin{aligned} \mathsf{Adv}(\mathcal{R}) &\leq \mathsf{Succ}_{\Gamma}^{\mathsf{Game1}}(\mathcal{A}) \\ &\leq \mathsf{Succ}_{\Gamma}^{\mathsf{Game0}}(\mathcal{A}) + \Delta \\ &\leq \mathsf{Succ}(NM - CPA[\Gamma])(\mathsf{Succ}(OW - CCA[\Gamma])) + \Delta \end{aligned}$$

since Δ is negligible, then under the assumption of Γ being NM-CPA (OW-CCA) secure, the advantage of $\mathcal{R}$ is also negligible. □

Positive results

One can give an informal explanation to the result above as follows. It is well known that constructions obtained from the sign_then_encrypt paradigm are not *strongly unforgeable*. I.e. a polynomial adversary is able to produce, given a valid confirmer signature on a certain message, another valid confirmer signature on the same message without the help of the signer. Indeed, given a valid confirmer signature on a message, an attacker can request its corresponding digital signature from the selective conversion oracle, then he encrypts it under the confirmer public key and obtains a new confirmer signature on the same message. Therefore, any reduction $\mathcal{R}$ from the invisibility of the construction to the security of the underlying encryption scheme will need more than a list of records maintaining the queried messages along with the corresponding confirmer and digital signatures. Thus the insufficiency of notions like IND-CPA. In [Camenisch & Michels, 2000], the authors stipulate that the given reduction would need a decryption oracle (of the encryption scheme) in order to handle the queries made by the INV-CMA attacker $\mathcal{A}$, which makes the invisibility of the construction rest on the IND-CCA security of the encryption scheme. In our work, we remark that the queries made by $\mathcal{A}$ are not completely uncontrolled by $\mathcal{R}$. In fact, they are encryptions of some data already released by $\mathcal{R}$, provided the digital signature scheme is strongly unforgeable, and thus known to him. Therefore, a plaintext checking oracle suffices to handle those queries.

Theorem 4.7. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction given above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if the underlying digital signature is (t, ϵ', q_s)-SEUF-CMA secure and the underlying encryption scheme is $(t + q_s q_{sc}(q_{sc} + q_v), \epsilon \cdot (1 - \epsilon')^{(q_{sc}+q_v)}, q_{sc}(q_{sc} + q_v))$-IND-PCA secure.*

Proof. Let $\mathcal{A}$ be an attacker that $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA breaks the invisibility of the above confirmer signature, believed to use a (t, ϵ', q_s)-SEUF-CMA secure signature scheme. We will construct an algorithm $\mathcal{R}$ that IND-PCA breaks the underlying encryption scheme as follows:

[Key generation] $\mathcal{R}$ will get the public parameters of the target encryption scheme from his challenger, that are $\Gamma.\mathsf{pk}$, $\Gamma.\mathsf{encrypt}$, and $\Gamma.\mathsf{decrypt}$. Then, he will choose an appropriate signature scheme Σ with parameters $\Sigma.\mathsf{pk}$, $\Sigma.\mathsf{sk}$, $\Sigma.\mathsf{sign}$, and $\Sigma.\mathsf{verify}$.

[$\mathsf{confirmedSign}$ **queries]** For a signature query on a message m. $\mathcal{R}$ first computes a (digital) signature σ on m using his secret key $\Sigma.\mathsf{sk}$. Then, he encrypts σ and outputs the result to $\mathcal{A}$. Besides, $\mathcal{R}$ issues a ZK proof of knowledge of σ that satisfies the equation defined by $\Sigma.\mathsf{verify}$. Finally, $\mathcal{R}$ will maintain a list $\mathcal{L}$ of the queries (messages), the corresponding digital signatures and finally the signatures he issued. $\mathcal{R}$ will proceed in this way for *each query* and not only *each new query*.

[$\mathsf{convert}$ **queries]** For a putative confirmer signature μ on m, $\mathcal{R}$ will look up the list $\mathcal{L}$. We note that each record of $\mathcal{L}$ comprises three components : (1) the queried message m_i (2) σ_i corresponding to a digital signature on m_i (3) $\Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(\sigma_i) = \mu_i$, which corresponds to the confirmer signature issued on m_i. If no record having as first component the message m appears in $\mathcal{L}$, then $\mathcal{R}$ will output $\perp$. Otherwise, let t be the number of records having as first component the message m. $\mathcal{R}$ will invoke the plaintext checking oracle (PCA) furnished by his own challenger on (σ_i, μ), for $1 \leq i \leq t$, where σ_i corresponds to the second component of such records. If the PCA oracle identifies μ as a valid encryption of some σ_i, $1 \leq i \leq t$, then $\mathcal{R}$ will return σ_i, otherwise he will return $\perp$. This simulation differs from the real one when the signature μ is valid and was not obtained from the signing oracle. We note that the only ways to create a valid confirmer signature without the help of $\mathcal{R}$ consist in either encrypting a digital signature obtained from the conversion oracle or coming up with a new fresh pair of message and corresponding signature (m, μ). $\mathcal{R}$ can handle the first case using his PCA oracle and list of records $\mathcal{L}$. In the second case, we can distinguish two sub-cases: either m has not been queried to the signing oracle in which case the pair (m, μ) corresponds to an existential forgery on the confirmer signature scheme and thus to an existential forgery on the underlying digital scheme according to Theorem 4.1, or m has been queried to the signing oracle but $\Gamma.\mathsf{decrypt}(\mu)$ is not an output of the selective conversion oracle, which corresponds to a strong existential forgery on the underlying digital signature. Therefore, the probability that this scenario does not happen is at least $(1 - \epsilon')^{q_{sc}}$ because the underlying digital signature scheme is (t, ϵ', q_s)-SEUF-CMA secure by assumption.

[confirm/deny **queries]** $\mathcal{R}$ will proceed exactly as in the selective conversion with the exception of simulating the denial protocol instead of returning $\perp$, or the confirmation protocol instead of returning the converted digital signature. $\mathcal{R}$ can issue such proofs without knowing the private key of the encryption scheme using the rewinding technique (see Remark 1.9) because the protocols are zero knowledge and thus simulatable, or using designated verifier proofs [Jakobsson *et al.*, 1996] in a registered key model. Analogously, the probability that $\mathcal{A}$ does not query a valid signature he has not obtained from the signing oracle is at least $(1-\epsilon')^{q_v}$.

[Challenge phase] Eventually, $\mathcal{A}$ will output two challenging messages m_0 and m_1. $\mathcal{R}$ will then compute two signatures σ_0 and σ_1 on m_0 and m_1 respectively, which he gives to his own challenger. $\mathcal{R}$ will receive then the challenge $\mu^\star$, as the encryption of either σ_0 or σ_1, which he will forward to $\mathcal{A}$.

[Post challenge phase] $\mathcal{A}$ will continue issuing queries to the signing, confirmation/denial and selective conversion oracles and $\mathcal{R}$ can answer as previously. Note that in this phase, $\mathcal{A}$ is not allowed to query the selective conversion or the confirmation/denial oracles on $(m_i, \mu^\star)$, $i = 0, 1$. Also, $\mathcal{R}$ is not allowed to query his PCA oracle on $(\mu^\star, \sigma_i)$, $i = 0, 1$. If during the selective conversion or confirmation/denial queries made by $\mathcal{A}$, $\mathcal{R}$ is compelled to query his PCA oracle on $(\mu^\star, \sigma_i)$, $i = 0, 1$, he will simply output $\perp$ in case of a selective conversion query or simulate the denial protocol in case of a verification query. This differs from the real scenario when $\mu^\star$ is a valid confirmer signature on some message $m \notin \{m_0, m_1\}$, which corresponds to an existential forgery on the underlying signature scheme (σ_i will be a valid digital signature on m_0 or m_1 and on a message $m \notin \{m_0, m_1\}$). Again, this does not happen with probability at least $(1-\epsilon')^{q_{sc}+q_v}$.

[Final output] When $\mathcal{A}$ outputs his answer $b \in \{0, 1\}$, $\mathcal{R}$ will forward this answer to his own challenger. Therefore $\mathcal{R}$ will IND-PCA break the underlying encryption scheme with advantage at least $\epsilon \cdot (1-\epsilon')^{(q_v+q_{sc})}$, in time at most $t + q_s q_{sc}(q_v + q_{sc})$ after at most $q_{sc}(q_{sc} + q_v)$ queries to the PCA oracle.

□

Unfortunately, requiring the encryption scheme to be at least IND-PCA secure seems to impact negatively the efficiency of the construction as it excludes homomorphic schemes from use (a homomorphic encryption scheme cannot be IND-PCA secure). In fact, such schemes can be (as we will show later in this document) efficient decryption verifiable, i.e. they accept efficient ZK proofs of knowledge of the decryption of a given ciphertext. In the next section, we discuss an attempt to circumvent this problem.

Remark 4.5. *There exists a simpler way to exclude homomorphic encryption from the design which consists in proceeding as follows:*

First rule out the notions OW-CPA, IND-CPA and OW-PCA by remarking that ElGamal's encryption meets all those notions (under the CDH, DDH and GDH assumption resp.) but still cannot be used as an ingredient in the construction. In fact, ElGamal offers the possibility of, given a ciphertext, creating another ciphertext for the same message (multiply the first component by g^r, for some r, and the second one by y^r, where $(\mathsf{sk} = x, \mathsf{pk} = y = g^x)$ is the key pair of the scheme). Now, let (μ, m_0, m_1) be a challenge of an INV-CMA adversary $\mathcal{A}$. By construction, μ is an ElGamal encryption of some σ, which is a digital signature on either m_0 or m_1. By the argument above, $\mathcal{A}$ can create another confirmer signature μ', that is another encryption of σ, and that he can query (w.r.t. m_0 for example) to the selective conversion oracle and then answer his own challenge.
Next, deduce that the encryption scheme in constructions derived from the "encryption of a signature" paradigm must be at least OW-CCA or NM-CPA or IND-PCA secure in order to lead to secure constructions. Finally, conclude by the fact that a homomorphic scheme cannot be NM-CPA secure nor OW-CCA nor IND-PCA secure[1].
However, in order to determine the exact security needed to achieve secure constructions from the mentioned paradigm, there seems no known simpler way to exist than the study provided in this section.

4.2 An efficient construction from a variant of the paradigm

One attempt to circumvent the problem of *strong forgeability* of constructions obtained from the plain "encryption of a signature" paradigm can be achieved by binding the digital signature to its encryption. In this way, from a digital signature σ and a message m, an adversary cannot create a new confirmer signature on m by just re-encrypting σ. In fact, σ forms a digital signature on m and some data, say c, which uniquely defines the confirmer signature on m. Moreover, this data c has to be public in order to issue the confirmedSign/confirmation/denial protocols.

In this section, we propose a realization of this idea using hybrid encryption (the KEM/DEM paradigm). We also allow more flexibility without compromising the overall security by encrypting only one part of the signature and leaving out the other part, provided it does not reveal information about the key nor about the message.

[1]Let $\mathbb{E}$ be an encryption scheme such that $\forall m, m' \in \mathcal{M}$: $\mathbb{E}.\mathsf{encrypt}(m \star m') = \mathbb{E}.\mathsf{encrypt}(m) \circ \mathbb{E}.\mathsf{encrypt}(m')$, where $\mathcal{M}$ is the message space, encrypt is the encryption algorithm and finally $\star$ and $\circ$ are some group laws defined by $\mathbb{E}$ on the message and ciphertext spaces resp. Let c be the NM-CPA challenge. An adversary can simply choose a random message $m' \xleftarrow{R} \mathcal{M}$, encrypt it in c' and finally output $c \circ c'$ and the relation $R = \star m'$. Now, let c be a OW-CCA challenge, an adversary can choose again a random message $m' \xleftarrow{R} \mathcal{M}$, encrypt it in c' and then query $c \star c'$ to the decryption oracle. Let m" be the result, the adversary can simply output m" $\star\, m'^{-1}$ as the decryption of c (we assume that computing inverses in $\mathcal{M}$ is done efficiently). Similarly, a homomorphic scheme cannot be IND-PCA secure.

4.2.1 The construction

Let Σ be a digital signature scheme given by $\Sigma.\mathsf{keygen}$, which generates a key pair ($\Sigma.\mathsf{sk}$, $\Sigma.\mathsf{pk}$), $\Sigma.\mathsf{sign}$, and $\Sigma.\mathsf{verify}$. Let furthermore $\mathcal{K}$ be a KEM given by $\mathcal{K}.\mathsf{keygen}$, which generates a key pair ($\mathcal{K}.\mathsf{pk}$, $\mathcal{K}.\mathsf{sk}$), $\mathcal{K}.\mathsf{encap}$, and $\mathcal{K}.\mathsf{decap}$. Finally, we consider a DEM $\mathcal{D}$ given by $\mathcal{D}.\mathsf{encrypt}$ and $\mathcal{D}.\mathsf{decrypt}$.

We assume that any digital signature σ, generated using Σ on an arbitrary message m, can be efficiently transformed in a reversible way to a pair (s, r) where r reveals no information about m nor about $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$. I.e. there exists an algorithm that inputs a message m and a key pair $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$ and outputs a string statistically indistinguishable from r, where the probability is taken over the messages and the key pairs considered by Σ. This technical detail will improve the efficiency of the construction as it will not necessitate encrypting the entire signature σ, but only the message-key-dependent part, namely s. Finally, we assume that s belongs to the message space of $\mathcal{D}$.

In the rest of this section, we consider that the encapsulations generated by the KEM $\mathcal{K}$ are exactly κ-bit long, where κ is a security parameter. This can be for example realized by padding with zeros, on the left of the most significant bit of the given encapsulation, until the resulting string has length κ. Moreover, the operator $\|$ denotes the usual concatenation operation between two bit-strings. As a result, the first bit of m will always be at the $(\kappa+1)$-st position in $c\|m$, where c is a given encapsulation. Such a technical detail will play an important role in the unforgeability and invisibility of the construction.

The construction of confirmer signatures from Σ, $\mathcal{K}$, and $\mathcal{D}$ is given as follows.

Key generation (keygen**).** Call $\Sigma.\mathsf{keygen}$ and $\mathcal{K}.\mathsf{keygen}$ to generate, on input a security parameter κ, $\Sigma.\mathsf{sk}$, $\Sigma.\mathsf{pk}$, $\mathcal{K}.\mathsf{pk}$, and $\mathcal{K}.\mathsf{sk}$ respectively. Set the signer's key pair to $(\Sigma.\mathsf{sk}, \Sigma.\mathsf{pk})$ and the confirmer's key pair to $(\mathcal{K}.\mathsf{sk}, \mathcal{K}.\mathsf{pk})$.

ConfirmedSign ($\mathsf{confirmedSign}$**).** Fix a key k together with its encapsulation c. Then, compute a (digital) signature $\sigma = \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c\|m) = (s, r)$ on $c\|m$. Finally, output $\mu = (c, \mathcal{D}.\mathsf{encrypt}_k(s), r)$ and prove the knowledge of s, decryption of $(c, \mathcal{D}.\mathsf{encrypt}_k(s))$, which together with r forms a valid digital signature on $c\|m$ w.r.t. $\Sigma.\mathsf{pk}$. This proof is possible because the signer knows k and (s, r), and the last assertion defines an NP language which accepts a ZK proof system.

Confirmation/Denial protocol ($\mathsf{confirm/deny}$**).** To confirm (deny) a purported signature $\mu = (\mu_1, \mu_2, \mu_3)$, issued on a certain message m, the confirmer first computes $k = \mathcal{K}.\mathsf{decap}_{\mathcal{K}.\mathsf{sk}}(\mu_1)$ then calls $\Sigma.\mathsf{verify}$ on $(\mathcal{D}.\mathsf{decrypt}_k(\mu_2), \mu_3)$ and $\mu_1\|m$ using $\Sigma.\mathsf{pk}$. According to the result, the signer issues a ZK proof of knowledge of the decryption of (μ_1, μ_2) that, together with μ_3, passes (does not pass) the verification algorithm $\Sigma.\mathsf{verify}$. Again this proof is possible because the given assertions are either NP or co-NP statements and therefore accept a ZK proof system.

Selective conversion (convert**).** To convert a given signature $\mu = (\mu_1, \mu_2, \mu_3)$ issued on a certain message m, the confirmer first checks its validity. In case it is valid, the signer computes $k = \mathcal{K}.\mathsf{decap}_{\mathcal{K}.\mathsf{sk}}(\mu_1)$, outputs $(\mathcal{D}.\mathsf{decrypt}_k(\mu_2), \mu_3)$, and proves that k is the decapsulation of μ_1, otherwise he outputs $\perp$.

Theorem 4.8. *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the above construction is (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature scheme is (t, ϵ, q_s)-EUF-CMA secure.*

Proof. Let $\mathcal{A}$ be an attacker that (t, ϵ, q_s)-EUF-CMA breaks the existential unforgeability of the above construction. We will construct an adversary $\mathcal{R}$ that (t, ϵ, q_s)-EUF-CMA breaks the underlying digital signature scheme:

[Key generation] $\mathcal{R}$ gets the parameters of the signature scheme in question from his challenger. Then he chooses an appropriate KEM $\mathcal{K}$ and DEM $\mathcal{D}$ and asks $\mathcal{A}$ to provide him with the confirmer key pair $(\mathcal{K}.\mathsf{sk}, \mathcal{K}.\mathsf{pk})$. Finally, $\mathcal{R}$ fixes the above parameters as a setting for the confirmer signature scheme $\mathcal{A}$ is trying to attack.

[confirmedSign **queries]** For a signature query on a message m, $\mathcal{R}$ will first compute an encapsulation c together with its decapsulation k (using $\Gamma.\mathsf{pk}$). Then, he will request his challenger for a digital signature $\sigma = (s, r)$ on $c\|m$. Finally, he encrypts s in $\mathcal{D}.\mathsf{encrypt}_k(s)$, then outputs the confirmer signature $(c, \mathcal{D}.\mathsf{encrypt}_k(s), r)$ and proves in ZK its validity to $\mathcal{A}$.

[Final Output] Once $\mathcal{A}$ outputs his forgery $\mu^\star = (\mu_1^\star, \mu_2^\star, \mu_3^\star)$ on $m^\star$. $\mathcal{R}$ will compute the decapsulation of $\mu_1^\star$, say k. If $\mu^\star$ is valid then by definition $(\mathcal{D}.\mathsf{decrypt}_k(\mu_2^\star), \mu_3^\star)$ is a valid digital signature on $\mu_1^\star\|m^\star$. Thus, $\mathcal{R}$ outputs $(\mathcal{D}.\mathsf{decrypt}_k(\mu_2^\star), \mu_3^\star)$ and $\mu_1^\star\|m^\star$ as a valid existential forgery on Σ. In fact, if, during a query made by $\mathcal{A}$ on a message m^i , $\mathcal{R}$ is compelled to query his own challenger for a digital signature on $\mu_1^\star\|m^\star = \mu_1^i\|m^i$, then $m^\star = m^i$ (by construction), which contradicts the fact that $(\mu^\star, m^\star)$ is an existential forgery output by $\mathcal{A}$.

Note that there will be no need to simulate the confirmation/denial and selective conversion oracles since $\mathcal{A}$ knows $\mathcal{K}.\mathsf{sk}$ which allows the verification of the confirmer signatures. □

The following remark is vital for the invisibility of the resulting undeniable signatures.

Remark 4.6. *The previous theorem shows that existential unforgeability of the underlying digital signature scheme suffices to ensure existential unforgeability of the resulting construction. Actually, one can also show that this requirement on the digital signature (EUF-CMA security) guarantees that no adversary, against the construction, can come up with a valid confirmer signature $\mu = (\mu_1, \mu_2, \mu_3)$ (μ_1 is the encapsulation used to generate the confirmer signature μ) on a message m that has been queried before to the signing oracle but where μ_1 was never used to generate answers (confirmer signatures) to the signature queries.*

To prove this claim, we construct from such an adversary, say $\mathcal{A}$, an EUF-CMA adversary $\mathcal{R}$ against the underlying digital signature scheme, which runs in the same time and has the same

advantage as $\mathcal{A}$. In fact, $\mathcal{R}$ will simulate $\mathcal{A}$'s environment in the same way described in the proof of Theorem 4.8. When $\mathcal{A}$ outputs his forgery $\mu^\star = (\mu_1^\star, \mu_2^\star, \mu_3^\star)$ on a message m_i that has been previously queried to the signing oracle, $\mathcal{R}$ decrypts $(\mu_1^\star, \mu_2^\star)$ in $s^\star$, which by definition forms, together with $\mu_3^\star$, a valid digital signature on $\mu_1^\star \| m_i$. Since by assumption $\mu_1^\star$ was never used to generate confirmer signatures on the queried messages, $\mathcal{R}$ never invoked his own challenger for a digital signature on $\mu_1^\star \| m_i$. Therefore, $(s^\star, \mu_3^\star)$ will form a valid existential forgery on the underlying digital signature scheme.

Theorem 4.9. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction proposed above is $(t, \epsilon, q_s, q_v, q_{sc})$-SINV-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure digital signature, an INV-OT secure DEM and an $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1-\epsilon')^{q_v+q_{sc}})$-IND-CPA secure KEM.*

Proof. Let $\mathcal{A}$ be an attacker that $(t, \epsilon, q_s, q_v, q_{sc})$-SINV-CMA breaks our construction, assumed to use a (t, ϵ', q_s)-EUF-CMA secure digital signature and an INV-OT secure DEM. We will construct an algorithm $\mathcal{R}$ that $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1-\epsilon')^{q_v+q_{sc}})$-IND-CPA breaks the underlying KEM.

[Key generation] $\mathcal{R}$ gets the parameters of the KEM $\mathcal{K}$ from his challenger. Then, he chooses an appropriate INV-OT secure DEM $\mathcal{D}$ together with an (t, ϵ', q_s)-EUF-CMA secure signature scheme Σ.

[$\mathsf{confirmedSign}$ **queries]** For a signature query on m. $\mathcal{R}$ first fixes a session key k together with its decapsulation c using $\mathcal{K}.\mathsf{pk}$. Then he computes a (digital) signature $\sigma = (s, r)$ on $c\|m$ using $\Sigma.\mathsf{sk}$. Next, he encrypts s (using k) in $\mathcal{D}.\mathsf{encrypt}_k(s)$ and outputs to $\mathcal{A}$ the confirmer signature $(c, \mathcal{D}.\mathsf{encrypt}_k(s), r)$. Finally, he interacts with $\mathcal{A}$ in a ZK protocol where he proves that $(c, \mathcal{D}.\mathsf{encrypt}_k(s))$ is the encryption of some s which together with r forms a valid digital signature on $c\|m$ w.r.t. $\Sigma.\mathsf{pk}$. $\mathcal{R}$ will maintain a list $\mathcal{L}$ of the encapsulations c and keys k used to generate the confirmer signatures.

[$\mathsf{confirm}/\mathsf{deny}$ **queries]** For a signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a message m, R will look up the list $\mathcal{L}$. If a record having as first component the encapsulation μ_1, $\mathcal{R}$ will use the corresponding decapsulation, say k, to decrypt (μ_1, μ_2) in s. If (s, μ_3) is a valid digital signature on $c\|m$, $\mathcal{R}$ will run the confirmation protocol, otherwise, he will run the denial protocol. $\mathcal{R}$ can issue such proofs of knowledge, without knowing the private key of $\mathcal{K}$, using the rewinding technique because the protocols are zero knowledge, thus simulatable. In case μ_1 does not appear in any record of $\mathcal{L}$, $\mathcal{R}$ will issue the denial protocol.

This simulation differs from the real one when the signature $\mu = (\mu_1, \mu_2, \mu_3)$ on m is valid and μ_1 does not appear in any record of $\mathcal{L}$. We distinguish two cases: either m was never queried to the signing oracle, then (m, μ) would correspond to an existential forgery on the confirmer signature scheme, which would lead to an existential forgery on the underlying signature scheme by virtue of Theorem 4.8. The second case is when m has been previously queried to the signing oracle in which case (m, μ) would correspond to an existential forgery

on the underlying digital scheme thanks to Remark 4.6. Hence, the probability that both scenarios do not happen is at least $(1-\epsilon')^{q_v}$ because the underlying digital signature scheme is (t, ϵ', q_s)-EUF-CMA secure by assumption.

[convert **queries]** For a selective conversion query on $\mu = (\mu_1, \mu_2, \mu_3)$ and m, $\mathcal{R}$ will proceed as he would do in a verification (confirmation/denial) query with the exception of outputting the decryption of $(\mu_1, \mu_2,)$ together with μ_3 instead of simulating the confirmation protocol, or the symbol $\perp$ instead of the denial protocol. Again the probability that this simulation does not differ from the real execution of the algorithm is at least $(1-\epsilon')^{q_{sc}}$.

[Challenge] Eventually, $\mathcal{A}$ outputs a challenging message $m^\star$. $\mathcal{R}$ will use his challenge $(c^\star, k^\star)$ to compute a digital signature $(s^\star, r^\star)$ on $c^\star \| m^\star$. Then, he encrypts $s^\star$ using $k^\star$ and outputs $\mu^\star = (c^\star, \mathcal{D}.\mathsf{encrypt}_{k^\star}(s^\star), r^\star)$ to $\mathcal{A}$. Therefore, $\mu^\star$ is either a valid confirmer signature on $m^\star$ or an element indistinguishable from a random element in the (confirmer) signatures space ($k^\star$ is random according to Subsection 2.4.1 and the DEM is INV-OT secure). If $\mu^\star$, in the latter case, is a random element in the confirmer signatures space, then this complies with the scenario of a real attack. Otherwise, if $\mu^\star$ is *only indistinguishable from random*, then if the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an invisibility adversary in a real attack, then $\mathcal{A}$ can be easily turned into an attacker against the INV-OT security property of the DEM underlying the construction. To sum up, under the INV-OT assumption of the DEM underlying the construction, the challenge confirmer signature $\mu^\star$ is either a valid confirmer signature on $m^\star$ or a random element in the confirmer signature space.

[Post challenge phase] $\mathcal{A}$ will continue issuing queries to the signing, confirmation/denial and selective conversion oracles, and $\mathcal{R}$ can answer as previously. Note that in this phase, $\mathcal{A}$ might request the verification or selective conversion of a confirmer signature $(c^\star, -, -)$ on a message m_i. In this case, $\mathcal{R}$ will simply issue the denial protocol in case of a verification query, or the symbol $\perp$ in case of a selective conversion query. Following the same analysis as above, the probability that the simulation does not differ from the real execution is at least $(1-\epsilon')^{q_{sc}+q_v}$.

[Final output] When $\mathcal{A}$ outputs his answer $b \in \{0, 1\}$, $\mathcal{R}$ will forward this answer to his own challenger. Therefore $\mathcal{R}$ will $(t + q_s(q_v + q_{sc}), \epsilon \cdot (1-\epsilon')^{q_v+q_{sc}})$-IND-CPA break the KEM used in the construction.

□

Note that the strong unforgeability of the underlying signature scheme is not needed here to achieve invisibility. In fact, if the adversary can come up with another digital signature (s', r') on a given $c\|m$, then there is just one way to create the corresponding confirmer signature, namely encrypt s using $k = \mathcal{K}.\mathsf{decap}(c)$. Therefore, the reduction is able to handle a query requesting

the confirmation/denial or selective conversion of such a signature by just maintaining a list of the used encapsulations c and the corresponding decapsulations k.

4.3 Efficient instantiations

In this subsection, we define the classes of signatures/encryption schemes that yield efficient instantiations of the construction defined in the previous section (Section 4.2). The class of digital signatures we consider is very similar to the one defined by [Shahandashti & Safavi-Naini, 2008] in the context of designated verifier signatures, whereas the class of considered encryption schemes spotlights the importance of homomorphic encryption in the framework.

4.3.1 The class $\mathbb{S}$ of signatures

Definition 4.2. *$\mathbb{S}$ is the set of all digital signatures for which there exists a pair of efficient algorithms,* convert *and* retrieve*, where* convert *inputs a public key* pk*, a message m, and a valid signature σ on m (according to* pk*) and outputs the pair (s, r) such that:*

1. *r reveals no information about m nor about* pk*, i.e. there exists an algorithm* simulate *such that for every public key* pk *from the key space and for every message m from the message space, the output* simulate(pk, m) *is statistically indistinguishable from r.*

2. *there exists an algorithm* compute *that on the input* pk*, the message m and r, computes a description of an injective* one-way function $f : (\mathbb{G}, *) \rightarrow (\mathbb{H}, \circ_s)$*:*

   - *where $(\mathbb{G}, *)$ is a group and $\mathbb{H}$ is a set equipped with the binary operation $\circ_s$,*
   - *$\forall S, S' \in \mathbb{G}$: $f(S * S') = f(S) \circ_s f(S')$.*

   and an $I \in \mathbb{H}$, such that $f(s) = I$.

and retrieve *is an algorithm that inputs* pk*, m and the correctly converted pair (s, r) and retrieves the signature σ on m.*

The class $\mathbb{S}$ differs from the class $\mathbb{C}$, introduced in [Shahandashti & Safavi-Naini, 2008], in the condition required for the one way function f. In fact, in our description of $\mathbb{S}$, the function f should satisfy a homomorphic property, whereas in the class $\mathbb{C}$, f should only possess an efficient protocol for proving knowledge of a preimage of a value in its range. We show in Theorem 4.10 that signatures in $\mathbb{S}$ accept also efficient proofs for proving knowledge of preimages, and thus belong to the class $\mathbb{C}$. Conversely, one can claim that signatures in $\mathbb{C}$ are also in $\mathbb{S}$, at least from a practical point of view, since it is not known in general how to achieve efficient protocols for proving knowledge of preimages of f without having the latter item satisfy some homomorphic properties. It is worth noting that similar to the classes $\mathbb{S}$ and $\mathbb{C}$ is the class of signatures introduced

in [Goldwasser & Waisbard, 2004], where the condition of having an efficient protocol for proving knowledge of preimages is weakened to having only a *witness hiding* proof of knowledge. Again, although this is a weaker assumption on f, all illustrations of signatures in this wider class happen to be also in $\mathbb{C}$ and $\mathbb{S}$. Our resort to specify the homomorphic property on f will be justified later when describing the confirmation/denial protocols of the resulting construction. In fact, these protocols are concurrent composition of proofs of knowledge and therefore need a careful study as it is known that zero knowledge is not closed under concurrent composition. Finally, the class $\mathbb{S}$ encompasses most proposals that were suggested so far, e.g.

RSA-FDH [BELLARE & ROGAWAY, 1996]. The *Full Domain Hash RSA* is given by the key pair $(\mathsf{pk} = (N, e), \mathsf{sk} = d)$, where N is an RSA modulus and $ed = 1 \bmod \Phi(N)$. A valid signature σ on a message m satisfies $\sigma^e = H(m) \bmod N$, where H is public hash function. It is easy to see that:

$$(\sigma, \epsilon) \leftarrow \mathsf{convert}(\mathsf{pk}, m, \sigma) \text{ and } \sigma \leftarrow \mathsf{retrieve}(\mathsf{pk}, m, (\sigma, \epsilon)),$$

where ϵ is the empty string. The verification equation suggests the following one-way function and image:

$$f(x) = x^e \bmod N \text{ and } I = H(m).$$

Obviously f is homomorphic as $\forall x, y \in \mathbb{Z}_N^\times \colon f(xy) = f(x)f(y)$.

SCHNORR [SCHNORR, 1991]. Schnorr's signature operates in a group $(\mathbb{G}, \cdot)$ of order q and generated by g. The key pair is given by $(\mathsf{sk} = x, \mathsf{pk} = y = g^x)$. A signature on a message m is of the form $\sigma = (c, s)$ such that $c = H(g^s \cdot y^{-c}, m)$ for some random $c \in \mathbb{Z}_q$. We have:

$$(s, r = g^s y^{-c}) \leftarrow \mathsf{convert}(\mathsf{pk}, m, \sigma) \text{ and } \sigma = (H(r, m), s) \leftarrow \mathsf{retrieve}(\mathsf{pk}, m, (s, r)).$$

In fact, since $c \in \mathbb{Z}_q$ is random, then $r = g^s y^{-c}$ is also random in $\mathbb{G}$. The one-way function and image are given by:

$$f(x) = g^x \text{ and } I = r \cdot y^{h(r,m)}.$$

Obviously $\forall x, y \in \mathbb{Z}_N^\times \colon f(x + y) = f(x)f(y)$.

GHR [GENNARO *et al.*, 1999]. The GHR signature scheme is given by the private key $\mathsf{sk} = (p, q)$ and the public key $\mathsf{pk} = (p \cdot q = N, s)$ for some $s \in \mathbb{Z}_N^\times$. A signature σ on a message m satisfies the equation $\sigma^{\psi(m)} = s$, where ψ is a public hash function which maps arbitrary messages to prime numbers. We have:

$$(\sigma, \epsilon) \leftarrow \mathsf{convert}(\mathsf{pk}, m, \sigma) \text{ and } \sigma \leftarrow \mathsf{retrieve}(\mathsf{pk}, m, (\sigma, \epsilon)),$$

and

$$f(x) = x^{\psi(m)} \bmod N \text{ and } I = s.$$

BLS [BONEH *et al.*, 2004B]. The BLS signature operates in a bilinear group (denoted additively) $\mathbb{G} = \langle P \rangle$ of order q and is given by the key pair $(\mathsf{sk} = x, \mathsf{pk} = xP = Y)$. A signature σ on a message m satisfies $e(\sigma, P) = e(H(m), Y)$ where e is the bilinear pairing (with values in a group $\mathbb{H}$ denoted multiplicatively) underlying $\mathbb{G}$, and H is a public hash function with values in $\mathbb{G}$. We have:

$$(\sigma, \epsilon) \leftarrow \mathsf{convert}(\mathsf{pk}, m, \sigma) \text{ and } \sigma \leftarrow \mathsf{retrieve}(\mathsf{pk}, m, (\sigma, \epsilon)),$$

and

$$f(Q) = e(Q, P) \text{ and } I = e(H(m), Y).$$

It is obvious that f is one-way, otherwise the CDH problem is easy in $\mathbb{G}$ ($e(xP, yP) = e(xyP, P)$). Moreover $\forall P, Q \in G\colon f(Q + R) = f(Q)f(R)$ (bilinearity property of e).

Other examples in the class $\mathbb{S}$ are *Modified ElGamal [Pointcheval & Stern, 2000], Cramer-Shoup [Cramer & Shoup, 2000], Camenisch-Lysyanskaya-02 [Camenisch & Lysyanskaya, 2002]*, and most pairing-based signatures that have been proposed so far [Camenisch & Lysyanskaya, 2004; Boneh & Boyen, 2004; Zhang *et al.*, 2004; Waters, 2005] etc. The reason why $\mathbb{S}$ encompasses most digital signature schemes lies in the fact that a signature verification consists in applying a function f to the "vital" part of the signature in question, then comparing the result to an expression computed from the message underlying the signature, the "auxiliary" or "simulatable" part of the signature, and finally the public parameters of the signature scheme. The function f must be one-way, otherwise the signature scheme is trivially forgeable. Moreover, it (f) consists most of the time of an arithmetic operation (exponentiation, raising to a power, pairing computation, ...) which satisfies an easy homomorphic property.

Theorem 4.10. *The protocol depicted in Figure 4.1 is an efficient zero knowledge protocol for proving knowledge of preimages of the function f described in Definition 4.2.*

We first remark that the function f used in the definition of the class $\mathbb{S}$ induces a group law in $f(\mathbb{G})$ for the operation $\circ_s$. Moreover, we have $1_{f(\mathbb{G})} = f(1_{\mathbb{G}})$ and $\forall S \in \mathbb{G}\colon f(s)^{-1} = f(s^{-1})$.

Proof. For completeness, it is clear that if both parties follow the protocol, the prover will always be able to provide a proof that the verifier will accept.
For soundness, let us assume that the cheating prover $\tilde{P}$ is able to successfully carry out the above protocol without knowing s. That is, $\tilde{P}$, after having committed to a t_1, is able to answer the challenge b with a response z satisfying $f(z) = t_1 \circ_s f(s)^b$. Note that, for a fixed t, the last equation corresponds each challenge b to a unique response z (f is injective, and we assume that

Prover P		Verifier V
Choose $s' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(s')$	$\xrightarrow{t_1}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell \ \ (b \in \mathbb{N})$
	$\xrightarrow{z = s' * s^b}$	
		Verify that $f(z) = t_1 \circ_s I^b$

Figure 4.1: Proof system for membership to the language $\{s\colon f(s) = I\}$ Common input: I and Private input : s

2^ℓ is smaller than the order of the group $f(\mathbb{G})$, which is equal to the order of $\mathbb{G}$). Thus, since f is one-way, $\tilde{P}$ needs to guess b correctly beforehand in order to provide an accepting answer; $\tilde{P}$ will first choose $z \xleftarrow{R} \mathbb{G}$, then computes $t_1 = f(z) \circ_s (f(s)^{-1})^b$ and sends it as a commitment in the first step of the protocol, when he receives the correctly guessed b, he will simply answer with z. This results in a soundness error equal to $2^{-\ell}$, which corresponds to the probability of correctly guessing the challenge b.

To prove that the protocol is ZK, we provide the following simulator:

1. Generate uniformly a random challenge $b' \xleftarrow{R} \{0,1\}^\ell$. Choose a random $z \xleftarrow{R} \mathbb{G}$, compute $t_1 = f(z) \circ_s (f(s)^{-1})^b$ and sends it to the verifier.
2. Get b from the verifier.
3. If $b = b'$, the simulator sends back z. Otherwise, it goes to Step 2 (*rewinds* the verifier).

The prover's first message in the protocol is a random value t_1 in $f(\mathbb{G})$, and so is the simulator's. Moreover, the distributions of the responses of the prover and of the simulator are again identical. Finally, we observe that the simulator runs in expected time 2^ℓ since the probability of not rewinding the verifier is:

$$\begin{aligned}
\Pr[b = b'] &= \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i, b' = b_i] \\
&= \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i] \Pr[b' = b_i] \\
&= 2^{-\ell} \sum_{b_i \in \{0,1\}^\ell} \Pr[b = b_i] \\
&= 2^{-\ell}
\end{aligned}$$

Adjusting ℓ to a factor logarithmic in the security parameter ensures that the simulator will run in expected polynomial time.

□

4.3.2 The class $\mathbb{E}$ of encryption schemes

Definition 4.3. $\mathbb{E}$ *is the set of encryption schemes* Γ, obtained from the KEM/DEM paradigm, *that have the following properties:*

1. *The message space is a group* $\mathcal{M} = (\mathbb{G}, *)$ *and the ciphertext space* $\mathcal{C}$ *is a set equipped with a binary operation* $\circ_e$.

2. *Let* $m \in \mathcal{M}$ *be a message and* c *its encryption with respect to a key* pk. *On the common input* m, c, *and* pk, *there exists an efficient zero knowledge proof of* m *being the decryption of* c *with respect to* pk. *The private input of the prover is either the private key* sk, *corresponding to* pk, *or the randomness used to encrypt* m *in* c *(the randomness which is input to the KEM encapsulation algorithm).*

3. $\forall m, m' \in \mathcal{M}, \forall \mathsf{pk}\colon \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m * m') = \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$. *Moreover, given the randomness used to encrypt* m *in* $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ *and* m' *in* $\Gamma.\mathsf{encrypt}_{pk}(m')$, *one can deduce (using only the public parameters) the randomness used to encrypt* $m * m'$ *in* $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$.

Examples of encryption schemes in the above class are :

EL GAMAL [El Gamal, 1985] : ElGamal's encryption is a KEM/DEM-based encryption scheme. It operates in a group $(\mathbb{G}, \cdot) = \langle g \rangle$, and is given by the KEM key pair $(\mathsf{sk} = x, \mathsf{pk} = y = g^x)$. To encrypt a message $m \in \mathbb{G}$, one first fixes a key y^r together with its encapsulation g^r, then encrypts m by simply computing the product $m \cdot y^r$. The ciphertext consists of the pair (g^r, my^r). To decrypt a ciphertext (c, e), one first decapsulates c to obtain the key $k = c^x$, then retrieves $m = ek^{-1}$. Let $\circ_e$, the binary operation defined on $\mathbb{G} \times \mathbb{G}$, be the term-wise product:

$$\forall a, b, c, d \in G\colon (a, b) \circ_e (c, d) = (ac, bd).$$

ElGamal's encryption is clearly homomorphic since

$$\mathsf{encrypt}(m) \circ_e \mathsf{encrypt}(m') = (g^r, my^r) \circ_e (g^s, m'y^s) = (g^{r+s}, mm'y^{r+s}) = \mathsf{encrypt}(mm')$$

Moreover, one can compute the randomness used to encrypt $m \cdot m'$ in $\mathsf{encrypt}(m) \circ_e \mathsf{encrypt}(m')$ as the sum of the randomnesses used to generate $\mathsf{encrypt}(m)$ and $\mathsf{encrypt}(m')$ resp.
Finally, given a ciphertext and its corresponding plaintext, one can efficiently prove the correctness of this assertion. The private input of the prover is either the randomness used to produce the ciphertext, or the private key of the scheme. This proof is often called in the literature the proof of equality of two discrete logarithms. It was first provided in [Chaum & Pedersen, 1993]. Figure 4.2 depicts such a proof.

Prover		Verifier
Choose $t \xleftarrow{R} \mathbb{Z}$		
Compute $h_1 = g^t$		
Compute $h_2 = y^t$		
	$\xrightarrow{(h_1, h_2)}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell (b \in \mathbb{N})$
	$\xrightarrow{s = t + xb}$	
		Verify that $g^s = h_1 e_1^b$ and $e_1^s = h_2 e_2^b$

Figure 4.2: Proof system for $\{(e_1, e_2) \colon e_1 = g^x \ \wedge \ e_2 = y^x\}$ Common input: (e_1, e_2, y, g) and Private input: x

BBS [Boneh *et al.*, 2004a] : It consists of the following algorithms:

- setup. We consider a bilinear group $(\mathbb{G}, +)$, with prime order d, generated by P.
- keygen. Probabilistically generate two secret values $x_1, x_2 \in \mathbb{Z}_d^\times$ and compute $X_1 = x_1 P$ and $X_2 = x_2 P$. Set the private key to $\mathsf{sk} = (x_1, x_2)$ and the public key to $\mathsf{pk} = (X_1, X_2)$.
- encrypt. Let $m \in \mathbb{G}$ be a message. Generate a random nonce $(a, b) \in \mathbb{Z}_d^2$ and compute the *session key* $k = (a + b)P$ and its *encapsulation* $c = (aX_1, bX_2)$. The ciphertext corresponding to m is $(c, k + m)$.
- decrypt. Given the private key sk and the element $(c, k+m)$, where $c = (aX_1, bX_2)$, compute k as $k = x_1^{-1} a X_1 + x_2^{-1} b X_2$. Then recover m from $k + m$.

The BBS scheme is IND-CPA secure under the decision linear assumption (Definition 2.5). Moreover, it is evident that this scheme satisfies the homomorphic properties announced in Definition 4.3. Finally, the proof that a given BBS ciphertext c decrypts to some message m is simply the proof of equality of two discrete logarithms: the discrete logarithm of $e(aX_1, bX_2)$ in base $e(kP, X_2)$, and the discrete logarithm of X_1 in base P, where e is the pairing underlying the group $\mathbb{G}$.

Finally, the Paillier [Paillier, 1999] encryption scheme cannot be viewed as an instance of this class as it is not based on the KEM/DEM paradigm.

Theorem 4.11. *Let Γ be a OW-CPA secure encryption scheme from the above class $\mathbb{E}$. Let furthermore c be an encryption of some message under some public key* pk*. The protocol depicted in Figure 4.3 is a zero knowledge proof of knowledge of the decryption of c.*

The proof is similar to that of Theorem 4.10.

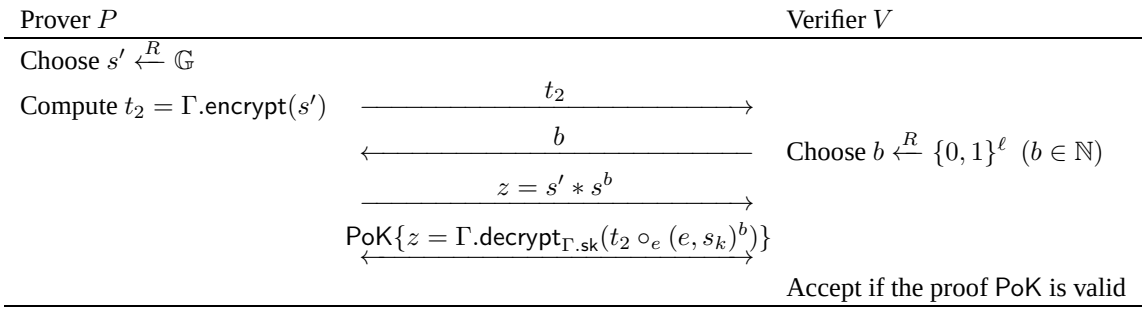

Figure 4.3: Proof system for membership to the language $\{(e, s_k) \colon \exists m : m = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(e, s_k)\}$ Common input: $(e, s_k, \Gamma.\mathsf{pk})$ and Private input: $\Gamma.\mathsf{sk}$ or randomness encrypting m in (e, s_k)

Proof. To prove this theorem, we first remark that the encryption algorithm, with respect to a given public key pk, induces a group law in the ciphertext space $\mathcal{C}$.

Completeness is straightforward. Soundness is again easy. In fact, we note that for a fixed commitment t_2, to each challenge b, corresponds a unique response z (we always assume that 2^ℓ is smaller than the order of the ciphertext space), namely the plaintext of the ciphertext $t_2 \circ_e (e, s_k)^b$. Thus, provided the encryption scheme Γ is one way, a cheating prover $\tilde{P}$ must guess correctly the challenge b in order to be able to carry out the protocol; i.e. he must choose $z \xleftarrow{R} \mathbb{G}$, then computes the commitment $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(z) \circ_e (e, s_k)^{-b}$ and sends it as the first message. Once $\tilde{P}$ receives the correctly guessed challenge, he will respond with z. We conclude that, provided PoK is sound, the soundness error probability of the protocol is at most $2^{-\ell}$.

For the zero-knowledgeness, we describe the following simulator:

1. Generate uniformly a random challenge $b' \xleftarrow{R} \{0,1\}^\ell$. Choose a random $z \xleftarrow{R} \mathbb{G}$, compute $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(z) \circ_e (e, s_k)^{-b}$ and send it to the verifier.

2. Get b from the verifier.

3. If $b = b'$, the simulator sends back z and simulates the proof PoK for z being the decryption of $t_2 \circ_e (e, s_k)^b$ (this proof is simulatable since it is zero knowledge by assumption). Otherwise, it goes to Step 2 (*rewinds* the verifier).

The prover's first message is always an encryption of a random value, and so is the first message of the simulator. Since b' is chosen uniformly at random from $\{0,1\}^\ell$, then, the probability that the simulator does not rewind the verifier is $2^{-\ell}$, and thus the simulator runs in expected polynomial time if ℓ is logarithmic in the security parameter. Finally, the distribution of the answers of the prover and of the simulator is again the same. We conclude that above proof is perfectly zero knowledge.

□

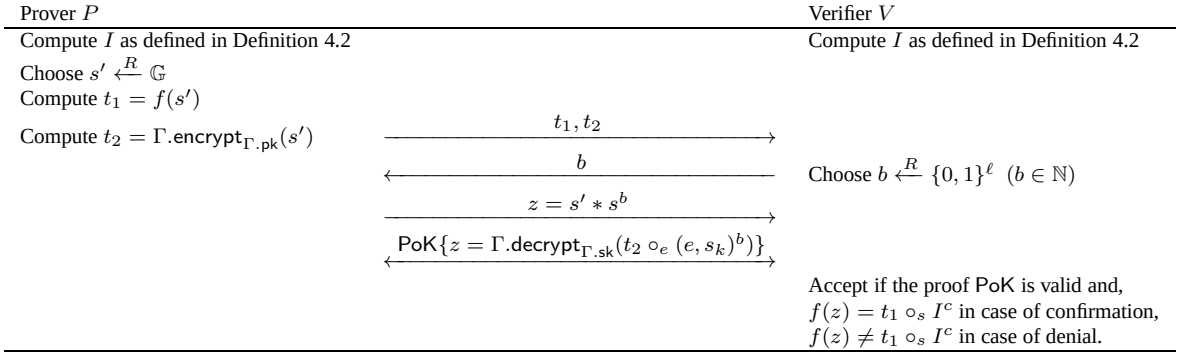

Figure 4.4: Proof system for membership to the language $\{(e, s_k, r) \colon \exists s \ : \ s = \Gamma.\mathsf{decrypt}(e, s_k) \wedge \Sigma.\mathsf{verify}(\mathsf{retrieve}(s, r), m \| e) = (\neq) 1\}$ Common input: $(e, s_k, r, \Sigma.\mathsf{pk}, \Gamma.\mathsf{pk})$ and Private input: $\Gamma.\mathsf{sk}$ or randomness encrypting s in (e, s_k)

4.3.3 The confirmation/denial protocols

We combine an EUF-CMA secure signature scheme $\Sigma \in \mathbb{S}$ and a encryption scheme $\Gamma \in \mathbb{E}$, where the underlying KEM $\mathcal{K}$ and DEM $\mathcal{D}$ are IND-CPA and INV-OT secure respectively, in the way described in Subsection 4.2.1. Namely we first compute an encapsulation e together with its corresponding key k. Then compute a signature σ on e concatenated with the message to be signed. Finally convert σ to (s, r) using the convert algorithm described in Definition 4.2 and encrypt s in $s_k = \mathcal{D}.\mathsf{encrypt}_k(s)$ using k. The resulting confirmer signature is (e, s_k, r). We describe in Figure 4.4 the confirmation/denial protocols corresponding to the resulting construction. Note that the confirmation protocol can be also run by the signer who wishes to confirm the validity of a just generated signature.

Remark 4.7. *The prover in Figure 4.4 is either the confirmer of the signature (e, s_k, r) who can run the above protocols with the knowledge of his private key, or the signer who wishes to confirm the validity of a just generated signature (during the* confirmedSign *protocol). In fact, with the knowledge of the randomness used to encrypt s in (e, s_k), where (s, r) is the converted pair obtained from $\sigma = \Sigma.\mathsf{sign}(m \| e)$, the signer can issue the above confirmation protocol thanks to the properties satisfied by Γ.*

Theorem 4.12. *The confirmation protocol (run by either the signer on a just generated signature or by the confirmer on any signature) described in Figure 4.4 is a proof of knowledge with perfect zero knowledge.*

Proof. The confirmation protocol depicted in Figure 4.4 is a parallel composition of the proofs depicted in Figures 4.1 and 4.3. Therefore completeness and soundness follow as a direct consequence from the completeness and soundness of the underlying proofs (see [Goldreich, 2001]).

To prove that the protocol is ZK, we provide the following simulator (for one execution):

1. Generate $b' \in_R \{0,1\}^\ell$. Choose $z \in_R \mathbb{G}$ and send $t_1 = f(z) \circ_s I^{-b}$ and $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(z) \circ_e (e, s_k)^{-b}$ to the verifier.

2. Get b from the verifier. If $b = b'$, it sends z and simulates the proof PoK of z being the decryption of $t_2 \circ_e (e, s_k)^b$ (this proof is simulatable since it is by assumption ZK). If $b \neq b'$, it goes to Step 1.

The prover's first message is an encryption of a random value $s' \in_R \mathbb{G}$, in addition to $f(s')$, and so is the simulator's first message (encryption of $z * s^{-b}$ and $f(z * s^{-b})$ where z is random). Therefore the distributions of the prover's and of the simulator's messages are the same in the first round of the proof. Moreover, the simulator runs in an expected polynomial time (we assume ℓ is logarithmic in the security parameter). Finally, the distribution of the prover's message in the third round is also similar to that of the simulator's. We conclude that the confirmation protocol is ZK. □

Theorem 4.13. *The denial protocol described in Figure 4.4 is a proof of knowledge with computational zero knowledge if the underlying encryption scheme is IND-CPA-secure.*

Proof. With the standard techniques, we prove that the denial protocol depicted in Figure 4.4 is complete and sound. Similarly, we provide the following simulator to prove the ZK property.

1. Generate $b' \in_R \{0,1\}$. Choose $z \in_R \mathbb{G}$ and a random $t_1 \in_R f(\mathbb{G})$ and $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(z) \circ_e (e, s_k)^{-b}$.

2. Get b from the verifier. If $b = b'$, it sends z and simulates the proof PoK of z being the decryption of $t_2 \circ_e (e, s_k)^b$. If $b \neq b'$, it goes to Step 1.

The prover's first message is an encryption of a random value $s' \in_R \mathbb{G}$, in addition to $f(s')$. The simulator's first message is an encryption of a random value $z * s$ and the element $t_1 \in_R f(\mathbb{G})$ (independent of z). Distinguishing these two cases is at least as hard as breaking the IND-CPA security of the underlying encryption scheme. In fact, if the verifier is able to distinguish these two cases, it can be easily used to break the encryption scheme in the IND-CPA sense. Therefore, under the assumption of the IND-CPA security of the encryption scheme, the simulator's and prover's first message distributions are indistinguishable. Moreover, the simulator runs in expected polynomial time, since the number of rewinds is 2^ℓ. Finally, the distributions of the prover's and the simulator's messages in the last round are again, by the same argument, indistinguishable under the IND-CPA security of the encryption scheme. □

Remark 4.8. *In case of confirmer signatures, ZK closedness under concurrent composition might be a desired property as it is natural to assume a confirmer (or a signer) involved in the confirmation/denial (or confirmedSign) of several signatures with several verifiers. Fortunately, there exists a result [Damgård, 2000] that shows a wide range of known zero knowledge protocols, for instance those provided in this chapter, to be modifiable with negligible loss of efficiency to preserve zero knowledgeness under concurrent composition.*

4.3.4 Comparisons and possible extentions

Sign_then_encrypt variants. The construction presented in Section 4.2 improves the plain paradigm [Camenisch & Michels, 2000] as it weakens the assumption on the underlying encryption scheme from being IND-CCA secure to only being IND-CPA secure. This impacts positively the efficiency of the construction from many sides. In fact, the resulting signature is shorter and its generation cost is smaller, since IND-CPA encryption schemes are simpler and allow faster encryption and shorter ciphertexts than IND-CCA ones. An illustration is given by ElGamal's encryption and its IND-CCA variant, namely Cramer-Shoup's encryption where the ciphertexts are at least twice longer than ElGamal's ciphertexts. Also, there is a multiplicative factor of at least two in favor of ElGamal's encryption/decryption cost. Moreover, the confirmation/denial protocols are rendered more efficient by the allowance of homomorphic encryption schemes as shown in Section 4.3.3. Such encryption schemes were not possible to use before since a homomorphic scheme can never attain the IND-CCA security. Besides, even when the IND-CCA encryption scheme is decryption verifiable, e.g. Cramer-Shoup, the involved protocols are much more expensive than those corresponding to their IND-CPA variant: in case of ElGamal, this protocol amounts to a proof of equality of two discrete logarithms. The construction achieves also better performances than the proposal of [Goldwasser & Waisbard, 2004], where the confirmer signature comprises k commitments and $2k$ IND-CCA encryptions, where k is the number of rounds used in the confirmation protocol. Moreover, the denial protocol presented in [Goldwasser & Waisbard, 2004] suffers the resort to proofs of general NP statements (where the considered encryption is IND-CCA). The same remark applies to the construction of [Wikström, 2007] where both the confirmation and denial protocols rely on proofs of general NP statements.

Commitment-based constructions. Our construction does not use the ROM, unlike the constructions in [Michels & Stadler, 1998; Wang *et al.*, 2007]. Moreover, it enjoys the strongest notion of invisibility (SINV-CMA) which captures both invisibility as defined in [Camenisch & Michels, 2000], and anonymity as defined in [Galbraith & Mao, 2003] and which can be an important requirement for confirmer signatures in some settings. Unfortunately, many of the efficient generic constructions are not anonymous. In fact, constructions like [Michels & Stadler, 1998; Gentry *et al.*, 2005; Wang *et al.*, 2007] have a confirmer signature containing a commitment on the message to be signed and a valid digital signature on this commitment. Therefore, such constructions leak always a part of the signing key, namely the public key of the underlying digital signature. More precisely, an anonymity attacker $\mathcal{A}$, will get two public keys and a confirmer signature on a given message and has to tell the key under which the confirmer signature was created. To answer such a challenge, $\mathcal{A}$ will simply check the validity of the digital signature on the commitment (both are part of the confirmer signature) with regard to one public key (the confirmer signature public key includes the public key of the underlying digital signature). The result of such a verification is sufficient for $\mathcal{A}$ to conclude in case the two confirmer public keys do not share the same public key for the digital signature scheme.

The upshot is, our construction achieves both maximal security (strong invisibility) without random oracles, and efficiency in terms of the signature length, generation, confirmation/denial, and conversion cost. Moreover, the construction readily extends to *directed signatures* [Lim & Lee, 1993] or *undeniable confirmer signatures* [Nguyen *et al.*, 1999] by simply having the confirmer share his private key with the signer. Furthermore, one can extend the analysis provided in this chapter to the other constructions instantiating the "encryption of a signature" paradigm, e.g. [Goldwasser & Waisbard, 2004; Wikström, 2007]. In fact, both constructions are not strongly unforgeable, thus the necessity of CCA or Δ-CCA security. To circumvent this problem, one can use similarly a encryption scheme derived from the hybrid encryption paradigm, and produce a signature on the message concatenated with the encapsulation. Hence, the resulting constructions will thrive on CPA or Δ-CPA security while conserving the same security, and thus will achieve better performances as we described above (short signature, small cost and many practical instantiations).

4.4 Conclusion

We provided a thorough analysis of the "encryption of a signature" paradigm. In fact, we set the necessary/minimal and sufficient assumptions on the building blocks in order to achieve unforgeable and invisible designated confirmer signatures under a chosen message attack. Next, we proposed a construction of confirmer signatures from a variant of the sign_then_encrypt paradigm whose invisibility rests on IND-CPA secure encryption schemes. Finally, we demonstrated the efficiency of our construction by explicitly giving the confirmation/denial protocols of the resulting signatures when instantiated with building blocks from a large class of signatures/encryption schemes. The next direction of research might be to check the necessity of the assumptions, in light of the previous study, required for the security of the new proposed framework or of the constructions that use commitment schemes.

Chapter 5

The "Signature of a Commitment" Paradigm

Abstract. Generic constructions of designated confirmer signatures follow one of the following two strategies; either produce a digital signature on the message to be signed, then encrypt the resulting signature, or produce a commitment on the message, encrypt the string used to generate the commitment, and finally sign the latter.
In this chapter, we revisit the second approach. In fact, efficient as the first approach is, it still applies only to a restricted class of signatures. This is clearly manifested in the constructions in the previous chapter which do not seem to be plausible with the PSS signature scheme [Bellare & Rogaway, 1996]. Our goal is to further improve the "commit then sign" method in terms of efficiency and security by allowing more efficient instantiations of the encryption and commitment schemes used as building blocks. Therefore, we first try to determine the exact security property needed in the encryption to achieve secure constructions. Our study infers the exclusion of a useful type of encryption from the design due to an intrinsic weakness in the paradigm. Next, we propose a simple method to remediate to this weakness and we get efficient constructions which can be used with *any* digital signature.
Parts of the results in this chapter will appear in the proceedings of ProvSec 2010 [El Aimani, 2010].

5.1 Analysis of the plain paradigm

We consider the construction of the plain "signature of a commitment" paradigm depicted in Subsection 3.3.2:

Setup (setup**).** Consider a digital signature scheme Σ, an encryption scheme Γ with labels, and a commitment scheme Ω.

Key generation (keygen**).** The signer key pair consists of $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$, corresponding to the key pair generated by $\Sigma.\mathsf{keygen}$, whereas the confirmer key pair consists of $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$ which corresponds to the key pair generated by $\Gamma.\mathsf{keygen}$.

ConfirmedSign (confirmedSign**).** To sign a message m, the signer first computes a commitment c on the message, then encrypts in e, under the label $m\|\Sigma.\mathsf{pk}$, the random string used for the commitment, say r, and finally, signs the commitment c using $\Sigma.\mathsf{sk}$. The confirmer signature consists of the triple $(e, c, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c))$. Next, the signer interacts with the verifier in a protocol where he proves in ZK the knowledge of r such that $r = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}, m\|\Sigma.\mathsf{pk}}(e)$ and $c = \Omega.\mathsf{commit}(m, r)$.

Confirmation/Denial protocol (confirm/deny**).** To confirm/deny a signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a given message m, the confirmer first checks whether μ_3 is a valid digital signature on μ_2 w.r.t. $\Sigma.\mathsf{pk}$, if so, he provides a concurrent ZK proof (using his private key $\Gamma.\mathsf{sk}$) of the equality/inequality of the decryption of μ_1 (w.r.t. the label $m\|\Sigma.\mathsf{pk}$) and the opening value of the commitment μ_2 w.r.t. m.

Verification (verify**).** The verification of a purported signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a given message m is achieved by first checking the validity of μ_3 w.r.t. to m as a digital signature, then checking the equality of the decryption of μ_1 (w.r.t. the label $m\|\Sigma.\mathsf{pk}$) and the opening value of the commitment μ_2 on m. This equality check can be performed by the signer, who has just generated μ, given the randomness used to create the ciphertext μ_1, or by the confirmer who can decrypt μ_1 using $\Gamma.\mathsf{sk}$.

Selective conversion (convert**).** Selective conversion of a signature $\mu = (\mu_1, \mu_2, \mu_3)$ is achieved by releasing the decryption of μ_1, in case μ is valid (the triple $(\Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(\mu_1), \mu_2, \mu_3)$ forms the converted signature), or the symbol $\perp$ otherwise.

Selective verification (verifyConverted**).** It is easy to see that the verification of converted signatures can be achieved by the algorithms $\Omega.\mathsf{open}$ and $\Sigma.\mathsf{verify}$.

The construction was shown, in Subsection 3.3.2, to be unforgeable and invisible in *the insider security model* if it uses a SEUF-CMA secure digital signature, an IND-CCA secure encryption and a secure commitment.

In the rest of this section, we prove that IND-PCA encryption schemes with labels are a minimal and sufficient requirement to obtain security for the confirmer, *in the outsider model*, if the underlying commitment scheme is secure, and the underlying signature is SEUF-CMA secure. Our study is similar to the one provided in the previous chapter (Subsection 4.1.2) which analyzes the plain "encryption of a signature" paradigm. Thus, we will first exclude OW-CCA secure encryption schemes with labels from use, which will rule out automatically OW-CPA and OW-PCA encryption schemes. We do this by using an efficient algorithm (a *meta-reduction*) which transforms the algorithm (*reduction*), reducing the invisibility of the confirmer signatures to the

OW-CCA security of the underlying encryption scheme, to an algorithm breaking the OW-CCA security of the same encryption scheme. Hence, such a result suggests that under the assumption of the underlying encryption scheme being OW-CCA secure, there exists no such a reduction, or if it (the encryption scheme) is not OW-CCA secure, such a reduction will be useless. Next, we exclude similarly NM-CPA encryption schemes from the design, which will rule out IND-CPA encryption. The next security notion that has to be considered is IND-PCA, which turns out to be sufficient to achieve invisibility. Likewise, our impossibility results are in a first stage partial in the sense that they apply only to *key-preserving* reductions, i.e. reductions which, trying to attack a certain property of an encryption scheme given by the public key pk, feed the invisibility adversary with the confirmer public key pk. Next, we extend the result to arbitrary reductions under some complexity assumptions on the encryption scheme in question.

5.1.1 Impossibility results

Lemma 5.1. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV-CMA adversary $\mathcal{A}$ against the above construction into a OW-CCA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that OW-CCA breaks the encryption scheme in question.*

As mentioned in the discussion above, the lemma claims that under the assumption of the underlying encryption scheme being OW-CCA secure, there exists no key-preserving reduction $\mathcal{R}$ that reduces OW-CCA breaking the encryption scheme in question to INV-CMA breaking the construction, or if there exists such an algorithm, the underlying encryption scheme is not OW-CCA secure, thus rendering such a reduction useless.

Proof. Let $\mathcal{R}$ be the key-preserving reduction that reduces the invisibility of the construction to the OW-CCA security of the underlying encryption scheme. We construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to OW-CCA break the same encryption scheme by simulating an execution of the INV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme $\mathcal{M}$ is trying to attack w.r.t. a public key $\Gamma.\mathsf{pk}$. $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key $\Gamma.\mathsf{pk}$. After $\mathcal{M}$ gets the label L on which $\mathcal{R}$ wishes to be challenged, he ($\mathcal{M}$) forwards it to his own challenger. Finally, $\mathcal{M}$ gets a challenge ciphertext c, that he forwards to $\mathcal{R}$. Note that $\mathcal{M}$ is allowed to query the decryption oracle on any pair (ciphertext,label) except on the pair (c, L). Thus, all decryption queries made by $\mathcal{R}$, which are by definition different from the challenge (c, L), can be forwarded to $\mathcal{M}$'s own challenger. At some point, $\mathcal{M}$, acting as an INV-CMA attacker against the construction, will output two messages m_0, m_1 such that $L \notin \{m_0 \| \Sigma.\mathsf{pk}, m_1 \| \Sigma.\mathsf{pk}\}$, where $\Sigma.\mathsf{pk}$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ gets as response a challenge signature $\mu^\star = (\mu_1^\star, \mu_2^\star, \mu_3^\star)$ which he is required to tell to which message it corresponds. Since the messages m_0 and m_1 were chosen such that the label under which the encryption $\mu_1^\star$ is created (either $m_0 \| \Sigma.\mathsf{pk}$ or $m_1 \| \Sigma.\mathsf{pk}$) is different from the challenge label L, $\mathcal{M}$ can query his decryption oracle on both pairs $(\mu_1^\star, m_0 \| \Sigma.\mathsf{pk})$ or

$(\mu_1^\star, m_1 \| \Sigma.\mathsf{pk})$. Results of such queries will enable $\mathcal{M}$ to open the commitment $\mu_2^\star$, and thus check the validity of the signature $\mu^\star$ w.r.t. one of the messages m_0 or m_1. Finally, when $\mathcal{R}$ outputs his answer, decryption of the challenge (c, L), $\mathcal{M}$ will simply forward this result to his challenger. □

Lemma 5.2. *Assume there exists a key-preserving reduction $\mathcal{R}$ that converts an INV-CMA adversary $\mathcal{A}$ against the above construction to an NM-CPA adversary against the underlying encryption scheme. Then, there exists a meta-reduction $\mathcal{M}$ that NM-CPA breaks the encryption scheme in question.*

Proof. Let $\mathcal{R}$ be a key-preserving reduction that reduces the invisibility of the construction to the NM-CPA security of its underlying encryption scheme. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to NM-CPA break the same encryption scheme by simulating an execution of the INV-CMA adversary $\mathcal{A}$ against the construction.

Let Γ be the encryption scheme with labels $\mathcal{M}$ is trying to attack. $\mathcal{M}$ launches $\mathcal{R}$ over Γ with the same public key, say $\Gamma.\mathsf{pk}$. $\mathcal{M}$, acting as the INV-CMA adversary against the construction, queries $\mathcal{R}$ on $m_0, m_1 \xleftarrow{R} \{0,1\}^\star$ for confirmer signatures. Then, he queries the resulting strings $\mu_0 = (\mu_0^1, \mu_0^2, \mu_0^3)$ and $\mu_1 = (\mu_1^1, \mu_1^2, \mu_1^3)$ (corresponding to the confirmer signatures on m_0 and m_1 respectively) for a selective conversion. Let r_0 and r_1 be the output decryption of μ_0^1 and μ_1^1 resp. (i.e. the randomnesses used generate the commitments μ_0^2 and μ_1^2 on m_0 and m_1 resp.). With overwhelming probability, we have $r_0 \neq r_1$ [1], and if it is not the case, $\mathcal{M}$ will repeat the experiment until he obtains two different r_0 and r_1. Then, $\mathcal{M}$ inputs $\mathcal{D} = \{r_0, r_1\}$ to his own challenger as a distribution probability from which the plaintexts will be drawn. Moreover, he chooses uniformly at random a bit $b \xleftarrow{R} \{0,1\}$ and outputs to his challenger the challenge label $m_b \| \Sigma.\mathsf{pk}$, where $\Sigma.\mathsf{pk}$ is the public key of the digital signature underlying the construction. $\mathcal{M}$ will receive as a challenge encryption $\mu_b^\star$. At that point, $\mathcal{M}$ will query $\mathcal{R}$ on the string $(\mu_b^\star, \mu_b^2, \mu_b^3)$ and the message m_b for a selective conversion. If the result of such a query is different from $\perp$, then, $\mu_b^\star$ is a valid encryption of the random string used to generate the commitment μ_b^2, namely r_b. $\mathcal{M}$ will then output to his challenger an encryption μ of $\overline{r_b}$ under the same challenge label $m_b \| \Sigma.\mathsf{pk}$, where $\overline{r_b}$ refers to the bit-complement of the element r_b, and the relation R: $R(r, r') = (r' = \overline{r})$. Otherwise, he will output an encryption of $\overline{r_{1-b}}$ (under the same challenge label) and the same relation R. Finally $\mathcal{M}$ aborts the game (stops simulating an INV-CMA attacker against the generic construction). □

Thus, when the considered notions are obtained from pairing a goal GOAL $\in \{\text{OW}, \text{IND}, \text{NM}\}$ and an attack model ATK $\in \{\text{CPA}, \text{PCA}, \text{CCA}\}$, we have

Theorem 5.3. *The encryption scheme underlying the above construction must be at least IND-PCA secure, in case the considered reduction is key-preserving, in order to achieve INV-CMA secure signatures.* □

[1] Actually, if $\mathcal{R}$ uses always the same string to produce the commitments, then the construction is clearly not invisible.

Similarly to the study in the previous chapter (Subsection 4.1.2), we generalize the above theorem to arbitrary reductions if the encryption scheme underlying the construction has a *non-malleable key generator*.

Theorem 5.4. *If the encryption scheme underlying the above construction has a non-malleable key generator, then it must be at least IND-PCA secure in order to achieve INV-CMA secure confirmer signatures.*

The proof is similar to that of Theorem 4.5. □

Remark 5.1. *Note that the above impossibility result holds true only when the considered notions are those obtained from pairing a security goal GOAL* $\in \{OW, IND, NM\}$ *and an attack model ATK* $\in \{CPA, PCA, CCA\}$. *Presence of other notions requires an additional analysis, however Lemmas 5.1 and 5.2 will still serve when there is a relation between the new notion and the notions NM-CPA and OW-CCA.*

5.1.2 Positive results

One way to explain the above result is to remark that the construction in question is not *strongly unforgeable*. In fact, an adversary $\mathcal{A}$, given a valid signature $\mu = (\mu_1, \mu_2, \mu_3)$ on a message m, can create another valid signature μ' on m without the help of the signer as follows: $\mathcal{A}$ will first request the selective conversion of μ to obtain the decryption of μ_1, say r, which he will re-encrypt in μ'_1 under the same label $m\|\Sigma.\mathsf{pk}$ ($\Sigma.\mathsf{pk}$ is the public key of the digital signature underlying the construction). Obviously, $\mu' = (\mu'_1, \mu_2, \mu_3)$ is also a valid confirmer signature on m that the signer did not produce, and thus cannot confirm/deny or convert without having access to a decryption oracle of the encryption scheme underlying the construction. This explains the insufficiency of notions like IND-CPA. However, we observe that an IND-CCA secure encryption is more than needed in this framework since a query of the type μ' is not completely uncontrolled by the signer. In fact, its first component μ'_1 is an encryption of some data already disclosed by the signer, namely r, and thus a plaintext checking oracle is sufficient to deal with such a query if the used digital signature is SEUF-CMA secure.

Theorem 5.5. *Given* $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ *and* $(\varepsilon, \epsilon') \in [0,1]^2$*, the construction given above is* $(t, \epsilon, q_s, q_v, q_{sc})$*-INV-CMA secure if it uses a* (t, ϵ', q_s)*-SEUF-CMA secure digital signature, an injective, binding and* (t, ϵ_h)*-hiding commitment, and a* $(t + q_s q_{sc}(q_{sc} + q_v), \frac{1}{2}(\epsilon + \epsilon_h) \cdot (1 - \epsilon')^{(q_{sc}+q_v)}, q_{sc}(q_{sc} + q_v))$*-IND-PCA secure encryption scheme with labels.*

Proof. Let $\mathcal{A}$ be an attacker against the construction. We will construct an attacker $\mathcal{R}$ against the underlying encryption scheme as follows.

[Parameter generation] $\mathcal{R}$ gets the parameters of the encryption scheme Γ from his challenger. Then he chooses a signature scheme Σ (along with a key pair $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$) and a suitable commitment scheme Ω. $\mathcal{R}$ sets the above entities as components of the construction $\mathcal{A}$ is trying to attack.

[confirmedSign **queries]** For a signature query on a message m_i, $\mathcal{R}$ computes a commitment c_i on m_i using a random string r_i, which he encrypts in e_i under the label $m_i \| \Sigma.\mathsf{pk}$, then he produces a digital signature σ_i on c_i using $\Sigma.\mathsf{sk}$. Next, he outputs $\mu_i = (e_i, c_i, \sigma_i)$ as a confirmer signature on m_i and a ZK proof of knowledge of the equality of the decryption of e_i and the string used in the commitment c_i. Such a proof is possible using the randomness t_i used to encrypt r_i in e_i. Finally, $\mathcal{R}$ adds the record $R_i = (m_i, t_i, r_i, e_i, c_i, \sigma_i)$ to a history list $\mathcal{L}$.

[confirm/deny **and** convert **queries]** To confirm/deny an alleged signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ on a message m_i, $\mathcal{R}$ will proceed as follows. First he checks the validity of the digital signature μ_i^3 on μ_i^2, in case it is invalid, he outputs $\perp$, otherwise he checks the list $\mathcal{L}$, if he finds a record R_i having as first field the message m_i, he will proceed to the next step, namely check whether the fourth field of R_i is equal to μ_i^1, if it is the case, $\mathcal{R}$ will issue a ZK proof of the equality/inequality of the decryption of μ_i^1 and the string used for the commitment μ_i^2. $\mathcal{R}$ can issue these proofs without the knowledge of $\Gamma.\mathsf{sk}$ using the rewinding technique (the proofs are ZK and thus simulatable) or by using the second field of R_i (randomness used to produce the encryption μ_i^1). Now, if R_i contains m_i in its first field, but its fourth field is different from μ_i^1, then $\mathcal{R}$ will check the next record R_j ($j > i$) having m_i in its first field and proceed in a similar fashion. Actually, if the message m_i is queried more than once, then it will occur in many records in $\mathcal{L}$. If $\mathcal{R}$ browses through all the records but none of them contains m_i and μ_i^1 in their first and fourth field resp., then for all the records R_i containing m_i in their first field, $\mathcal{A}$ will invoke his PCA oracle on the ciphertext μ_i^1 and the third fields of these records. If one of the queries yields "yes" as an answer, e.g. there exists a record $R_j = (m_i, t_j, r_j, e_j, c_j, \sigma_j)$ such that its third field r_j is a decryption of μ_i^1, then according to whether r_j is (is not) the opening value of the commitment μ_i^2 on m_i, $\mathcal{R}$ will issue a ZK proof of the equality (inequality) of the decryption of μ_i^1 and the string used for the commitment μ_i^2. Again such a proof is possible to issue using the rewinding technique (the value t_j cannot be used here because it was not used to encrypt r_j in μ_i^1). Finally, if no query to the PCA oracle yields the answer "yes", then $\mathcal{R}$ will issue the denial protocol, namely simulate a ZK proof, using the rewinding technique, of the inequality of the decryption of μ_i^1 and of the string used for the commitment μ_i^2.

Selective conversion is similarly carried out with the exception of issuing the decryption of μ_i^1 instead of the confirmation protocol and $\perp$ instead of the denial protocol.

The difference between the above simulation and the real execution of the algorithm is when the signature $\mu_i = (\mu_i^1, \mu_i^2, \mu_i^3)$ is valid, however, μ_i^1 is not an encryption of a string r_i already issued to $\mathcal{A}$ during a selective conversion query regarding the message m_i and a presumed signature on it. We distinguish two cases, either m_i was never queried for signature, in which case such a signature would correspond to an existential forgery on the construction and thus to an existential forgery on the underlying digital signature. Or, m_i was queried before for signature. Let $\mu_j = (\mu_j^1, \mu_j^2, \mu_j^3)$ be the output confirmer signature to such a query.

Since μ_i^1 is encryption of some r_i which was never used to generate signatures on m_i, then $\mu_i^2 \neq \mu_j^2$ (both are commitment on m_i with different random strings and Ω is injective). Thus, in this case (μ_i^2, μ_i^3) will correspond to an existential forgery on the underlying digital signature scheme. We conclude that the above simulation is indistinguishable from the real execution with probability at least $(1-\epsilon')^{q_v+q_{sc}}$, as the digital signature scheme underlying the construction is (t, ϵ', q_s)-SEUF-CMA secure by assumption.

[Challenge phase] At some point, $\mathcal{A}$ will output two messages m_0, m_1 to $\mathcal{R}$. The latter will then choose uniformly at random a bit $b \xleftarrow{R} \{0,1\}$, and two different random strings r_0 and r_1 from the corresponding space. $\mathcal{R}$ will output to his challenger the label $m_b \| \Sigma.\mathsf{pk}$ and the strings r_0, r_1. He receives then a ciphertext $e_{b'}$, encryption of $r_{b'}$, for some $b' \xleftarrow{R} \{0,1\}$. To answer his challenger, $\mathcal{R}$ will compute a commitment c_b on the message m_b using the string $r_{b''}$ where $b'' \xleftarrow{R} \{0,1\}$. Then, $\mathcal{R}$ will output $\mu = (e_{b'}, c_b, \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ as a challenge signature to $\mathcal{A}$. Two cases: either μ is valid confirmer signature on m_b (if $b' = b''$), or it is not a valid signature on neither m_0 nor m_1. If the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an INV-CMA attacker in a real attack, then , according to Lemma 3.5, $\mathcal{A}$ can be used to break the hiding property of Ω.

[Post challenge phase] $\mathcal{A}$ continues to issue queries and $\mathcal{R}$ continues to handle them as before. Note that at this stage, $\mathcal{R}$ cannot request his PCA oracle on $(e_{b'}, r_i)$, $i \in \{0,1\}$ under the label $m_b \| \Sigma.\mathsf{pk}$. $\mathcal{R}$ would need to query his PCA oracle on such a quantity if he gets a verification (conversion) query on a signature $(e'_b, c_b, -) \neq \mu$ and the message m_b. $\mathcal{R}$ will respond to such a query by simulating the denial protocol (output $\perp$). This simulation differs from the real algorithm when $(e'_b, c_b, -)$ is valid on m_b. Again, such a scenario won't happen with probability at least $(1-\epsilon')^{q_v+q_{sc}}$, because the query would form a strong existential forgery on the digital signature scheme underlying the construction.

[Final output] The rest of the proof follows in a straightforward way. Now, let $\mu = (e_{b'}, c_b,$-$\Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c_b))$ be the challenge signature. Let b_a be the bit output by $\mathcal{A}$. $\mathcal{R}$ will output b'' to his challenger in case $b = b_a$ and $1 - b''$ otherwise.

The advantage of $\mathcal{A}$ in such an attack is defined by

$$\epsilon = \mathsf{Adv}(\mathcal{A}) = \left| \Pr[b_a = b | b' = b''] - \frac{1}{2} \right|$$

We also have

$$\epsilon_h = \left| \Pr[b_a \neq b | b' \neq b''] - \frac{1}{2} \right|$$

We assume again without loss of generality that $\epsilon = \Pr[b_a = b | b' = b''] - \frac{1}{2}$ and $e_h = \Pr[b_a \neq b | b' \neq b''] - \frac{1}{2}$. The advantage of $\mathcal{R}$ is then given by

$$
\begin{aligned}
\mathsf{Adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}}\left[\Pr[b=b_a, b'=b''] + \Pr[b\neq b_a, b'\neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}}\left[\Pr[b=b_a|b'=b'']\Pr[b'=b''] + \Pr[b\neq b_a|b'\neq b'']\Pr[b'\neq b''] - \frac{1}{2}\right] \\
&= (1-\epsilon')^{q_v+q_{sc}}\left[\frac{1}{2}(\epsilon+\frac{1}{2}) + \frac{1}{2}(\epsilon_h+\frac{1}{2}) - \frac{1}{2}\right] \\
&= \frac{1}{2}(\epsilon+\epsilon_h)(1-\epsilon')^{q_v+q_{sc}}
\end{aligned}
$$

The last but one equation is due to the facts $\Pr[b' \neq b''] = \Pr[b' = b''] = \frac{1}{2}$ as $b'' \xleftarrow{R} \{0,1\}$, and to the fact that, in case $b' \neq b''$, the probability that $\mathcal{A}$ answers $1-b$ is $\frac{1}{2}$ greater than the advantage of the adversary in the game defined in Lemma 3.5, which is equal to ϵ_h.

□

5.2 An efficient construction from a variant of the paradigm

One simple way to eliminate the already mentioned weakness (strong forgeability) in signatures from the plain "signature of a commitment" technique consists in producing a digital signature on both the commitment and the encryption of the random string used in it. In this way, the attack discussed before Theorem 5.5 no longer applies, since an adversary will need to produce a digital signature on the commitment and the re-encryption of the random string used in it. Note that such a fix already appears in the construction of [Gentry *et al.*, 2005], however, it was not exploitable as the invisibility was considered in the insider model.

5.2.1 Construction

Let Σ be a signature scheme given by $\Sigma.\mathsf{keygen}$, that generates $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$, $\Sigma.\mathsf{sign}$, and $\Sigma.\mathsf{verify}$. Let further Γ denote an encryption scheme given by $\Gamma.\mathsf{keygen}$, that generates $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$, $\Gamma.\mathsf{encrypt}$, and $\Gamma.\mathsf{decrypt}$. We note that Γ does need to support labels in our construction. Finally let Ω denote a commitment scheme given by $\Omega.\mathsf{commit}$ and $\Omega.\mathsf{open}$. We assume that Γ produces ciphertexts of length exactly a certain κ. As a result, the first bit of c will always be at the $(\kappa+1)$-st position in $e\|c$, where e is an encryption produced by Γ . Such a technical detail will play an important role in the unforgeability and invisibility of the construction.

The construction of confirmer signatures from Σ, Γ, and Ω is given as follows.

Key generation. The signer key pair is $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$ and the confirmer key pair is $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$.

ConfirmedSign. On input a message m, the signer produces a commitment c on m using a random string r, encrypts this string in e, and then produces a digital signature $\sigma = \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(e\|c)$. Finally, the signer outputs $\mu = (e, c, \sigma)$ as a confirmer signature on m, and interacts with the verifier to prove in ZK the equality of the decryption of e and of the string used for the commitment c. This proof is possible using the randomness used to encrypt r in e.

Confirmation/Denial protocol. On a message m and an alleged signature $\mu = (\mu_1, \mu_2, \mu_3)$, the confirmer checks the validity of μ_3 on $\mu_1\|\mu_2$. In case it is not valid, he produces $\perp$. Otherwise, he computes the decryption r of μ_1 and checks $\mu_2 \stackrel{?}{=} \Omega.\mathsf{commit}(m, r)$, according to the result he interacts with the verifier to prove in ZK the equality/inequality of the decryption of μ_1 and of the string used to create μ_2.

Selective conversion. The confirmer proceeds as in the confirmation/denial protocol with the exception of issuing the decryption of μ_1 in case the signature is valid or the symbol $\perp$ otherwise.

5.2.2 Security analysis

First, we note that the security for the verifier and the non transferability of the confirmedSign, confirmation, and denial protocols are ensured by using zero knowledge proofs of knowledge. Furthermore, the construction is EUF-CMA secure and INV-CMA secure if the underlying components are secure.

Theorem 5.6. *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]^2$, the construction depicted above is (t, ϵ, q_s)-EUF-CMA secure if it uses a statistically binding commitment scheme and a (t, ϵ, q_s)-EUF-CMA secure digital signature scheme.*

Proof. (Sketch)

Let $\mathcal{A}$ be an EUF-CMA attacker against the construction. We construct an EUF-CMA attacker $\mathcal{R}$ against the underlying digital signature scheme as follows.

$\mathcal{R}$ gets the parameters of the digital signature from his attacker, and chooses a suitable encryption and commitment scheme. Simulation of the confirmedSign queries (on messages m_i) is done by first computing a commitment c_i on m_i using some random string r_i, then encrypting the string r_i in e_i and finally requesting the challenger for a digital signature σ_i on $e_i\|c_i$. The string (e_i, c_i, σ_i) is output to $\mathcal{A}$ along with a proof of equality of the decryption of e_i and of the opening value of c_i. Such a proof can be issued using the encryption scheme private key that $\mathcal{R}$ knows or the randomness used to encrypt r_i in e_i. Confirmation/denial and selective conversion queries can be perfectly simulated with the knowledge of the encryption scheme private key.

At some point, $\mathcal{A}$ will output a forgery $\mu^\star = (e^\star, c^\star, \sigma^\star)$ on some message $m^\star$, which was never queried before for signature. By definition, $\sigma^\star$ is a valid digital signature on $e^\star\|c^\star$. It will form

an existential forgery on the digital signature scheme if $e^\star \| c^\star$ was never queried before by $\mathcal{R}$ for a digital signature. Suppose there exists $1 \leq i \leq q_s$ such that $e^\star \| c^\star = e_i \| c_i$ where $\mu_i = (e_i, c_i, \sigma_i)$ was the output confirmer signature on the query m_i. Due to the special way the strings $e_i \| c_i$ are created, equality of the strings $e^\star \| c^\star$ and $e_i \| c_i$ implies equality of their suffixes (that start at the $(\kappa + 1)$-st position), namely $c^\star$ and c_i. This equality implies the equality of m_i and $m^\star$ since the used commitment is binding by assumption. Thus, $\mathcal{R}$ returns $(\sigma^\star, e^\star \| c^\star)$ as a valid existential forgery against the digital signature in question. □

Theorem 5.7. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction depicted above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if it uses an (t, ϵ', q_s)-SEUF-CMA secure digital signature, an injective, statistically binding, and (t, ϵ_h)-hiding commitment, and a $(t + q_s(q_v + q_{sc}), \frac{1}{2}(\epsilon + \epsilon_h)(1 - \epsilon')^{q_v + q_{sc}})$-IND-CPA secure encryption scheme.*

Proof. **[Parameter generation]** Simulation of the key generation is similar to the key generation in the proof of Theorem 5.5.

[confirmedSign **queries]** To sign a message m_i, $\mathcal{R}$ (the attacker against the encryption scheme) will proceed exactly as a real signer would do, with the exception of maintaining a list $\mathcal{L}$ of records that contains the strings used to form the commitments, their corresponding encryptions along with the random nonces used to produce these encryptions.

[confirm/deny **and** convert **queries]** For a verification query on (e_i, c_i, σ_i) and m_i (where σ_i is a valid digital signature on c_i), $\mathcal{R}$ will simulate the confirmation protocol (using the rewinding technique or the randomness used to encrypt the opening value of c_i in e_i) if the encryption e_i appears in at least one record of $\mathcal{L}$, or simulate the denial protocol otherwise. Selective conversion of a confirmer signature whose first field appears in the list is done by revealing the opening value of the commitment, otherwise such a confirmer signature is converted to $\perp$.

The difference between this simulation and the real execution of the algorithm manifest when a queried signature, say (e_i, c_i, σ_i), is valid but e_i was never used to generate confirmer signatures. We distinguish two cases, either the underlying message m_i has been queried previously on not. In the latter case, such a signature would correspond to an existential forgery on the construction, thus, to an existential forgery on the underlying digital signature. In the former case, let (e_j, c_j, σ_j) be the output signature to $\mathcal{A}$ on the message m_i. We have $e_i \| c_i \neq e_j \| c_j$ since $e_i \neq e_j$, and both e_i and e_j are the n-bit prefixes of $e_i \| c_i$ and $e_j \| c_j$ resp. We conclude that the adversary would have to compute a digital signature on a string for which he never had obtained a signature. Thus, the query would lead to an existential forgery on the underlying signature scheme. Since the latter is by assumption (t, ϵ', q_s)-SEUF-CMA secure, the probability that the simulation differs from the real execution is at least $(1 - \epsilon')^{q_v + q_{sc}}$.

[Challenge phase] Eventually, the adversary outputs two challenging messages m_0, m_1. $\mathcal{R}$ will then produce two different strings r_0, r_1 and hands them to his challenger. He gets as response a challenge ciphertext $e_{b'}$ on $r_{b'}$ for some $b' \in \{0,1\}$. $\mathcal{R}$ will choose two bits $b, b'' \stackrel{R}{\leftarrow} \{0,1\}$ and produce a commitment c_b on the message m_b using the string $r_{b''}$. Finally, he will produce a digital signature σ on $e_{b'} \| c_b$. The challenge confirmer signature is $\mu = (e_{b'}, c_b, \sigma)$. Note, that if $b' = b''$, the signature is valid on the message m_b, otherwise, it is invalid on both messages m_0 and m_1. Note also that if the advantage of $\mathcal{A}$ is non-negligibly different from the advantage of an INV-CMA attacker in a real attack, then, according to Lemma 3.5, $\mathcal{A}$ can be used to break the hiding property of Ω.

[Post challenge phase] The adversary will continue issuing his queries to $\mathcal{R}$ who will handle them as previously. Note that from now on and during the verification/conversion queries, the adversary may ask a query $(e_{b'}, c_b, -) \neq \mu$ on m_b. The probability that such a query is invalid is at least $(1-\epsilon')^{q_v+q_{sc}}$ since the digital signature scheme is (t, ϵ', q_s)-SEUF-CMA secure (if the underlying digital signature is not strongly unforgeable, then the adversary may come up with a new digital signature on $e_{b'} \| c_b$, say σ' which is different from σ, and then queries $(e_{b'}, c_b, \sigma')$ for verification or conversion; the result of such a query will enable him answer his challenge).

[Final output] At the end, the adversary outputs a bit b_a. Clearly the advantage of the adversary is $\epsilon = \Pr[b'' = b_a | b = b'] - \frac{1}{2}$. $\mathcal{R}$ will output b'' in case $b = b_a$ and $1 - b''$ otherwise.

Similarly, the advantage of $\mathcal{R}$ is:

$$\begin{aligned}
\mathsf{Adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a, b' = b''] + \Pr[b \neq b_a, b' \neq b''] - \frac{1}{2} \right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\Pr[b = b_a | b' = b''] \Pr[b' = b''] + \Pr[b \neq b_a | b' \neq b''] \Pr[b' \neq b''] - \frac{1}{2} \right] \\
&= (1-\epsilon')^{q_v+q_{sc}} \left[\frac{1}{2}(\epsilon + \frac{1}{2}) + \frac{1}{2}(\epsilon_h + \frac{1}{2}) - \frac{1}{2} \right] \\
&= \frac{1}{2}(\epsilon + \epsilon_h)(1-\epsilon')^{q_v+q_{sc}}
\end{aligned}$$

□

Remark 5.2. *Both Theorem 5.5 and Theorem 5.7 can be used with computationally binding commitments. The only issue is to have the formulation of both theorems complicated by further terms, e.g. ϵ_b, if we use a (t, ϵ_b)-binding commitment.*

5.2.3 Efficiency analysis

We show in this paragraph that requesting the encryption scheme to be only IND-CPA secure improves the efficiency of constructions from the "signature of a commitment" paradigm from many sides. First, it enhances the signature generation, verification, and conversion cost, as encryption and decryption are usually faster in IND-CPA secure encryption than in IND-CCA secure encryption (e.g. ElGamal vs Cramer-Shoup or Paillier vs Camenisch-Shoup). Next, we achieve also a shorter signature since ciphertexts produced using IND-CPA schemes are shorter than ciphertexts produced using IND-CCA secure encryption schemes. Finally, we allow homomorphic encryption in the design, which will render the confirmedSign/confirmation/denial protocols more efficient. In fact, in [Gentry *et al.*, 2005; Wang *et al.*, 2007], the signer/confirmer has to prove in ZK the equality/inequality of the decryption of an IND-CCA encryption and an opening value of a commitment scheme. Thus, the only efficient instantiation, that was provided, used Camenisch-Shoup's encryption and Pedersen's commitment. In the rest of this subsection, we enlarge the category of encryption/commitment schemes that yield efficient instantiations thanks to the allowance of homomorphic encryption in the design.

Definition 5.1. ***(The class $\mathbb{C}$ of commitments)*** *$\mathbb{C}$ is the set of all commitment schemes for which there exists an algorithm* compute *that on the input: the commitment public key* pk*, the message m and the commitment c on m, computes a description of an injective* one-way function $f : (\mathbb{G}, *) \rightarrow (\mathbb{H}, \circ_s)$ *where:*

- $(\mathbb{G}, *)$ *is a group and $\mathbb{H}$ is a set equipped with the binary operation $\circ_s$,*
- $\forall r, r' \in \mathbb{G}$*:* $f(r * r') = f(r) \circ_s f(r')$.

and an $I \in \mathbb{H}$, such that $f(r) = I$, where r is the opening value of c w.r.t. m.

It is easy to check that Pedersen's commitment scheme is in this class. Actually, most commitment schemes have this built-in property because it is often the case that the committer wants to prove efficiently that a commitment is produced on some message. This is possible if the function f is homomorphic as shown in Figure 5.1.

Theorem 5.8. *The protocol depicted in Figure 5.1 is an efficient zero knowledge protocol for proving knowledge of preimages of the function f described in Definition 5.1.*

The proof is similar to that of Theorem 4.10. □

For encryption, we use the same class $\mathbb{E}$ considered in Definition 5.2, with the exception of not requiring the encryption schemes to be derived from the hybrid encryption paradigm.

Definition 5.2. ***(The class $\mathbb{E}_2$ of encryption schemes)*** *$\mathbb{E}_2$ is the set of encryption schemes Γ that have the following properties:*

1. *The message space is a group $\mathcal{M} = (\mathbb{G}, *)$ and the ciphertext space $\mathcal{C}$ is a set equipped with a binary operation $\circ_e$.*

Prover P		Verifier V
Choose $r' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(r')$	$\xrightarrow{t_1}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^\ell \ \ (b \in \mathbb{N})$
	$\xrightarrow{z = r' * r^b}$	
		Verify that $f(z) = t_1 \circ_s I^b$

Figure 5.1: Proof system for membership to the language $\{r\colon f(r) = I\}$ Common input: I and Private input : r.

2. *Let $m \in \mathcal{M}$ be a message and c its encryption with respect to a key* pk*. On the common input* pk*, m, and c, there exists an efficient zero knowledge proof of m being the decryption of c with respect to* pk*. The private input of the prover is either the private key* sk*, corresponding to* pk*, or the randomness used to encrypt m in c.*

3. $\forall m, m' \in \mathcal{M}$, $\forall \mathsf{pk}\colon \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m * m') = \Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$. *Moreover, given the randomness used to encrypt m in $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m)$ and m' in $\Gamma.\mathsf{encrypt}_{pk}(m')$, one can deduce (using only the public parameters) the randomness used to encrypt $m * m'$ in $\Gamma.\mathsf{encrypt}_{\mathsf{pk}}(m) \circ_e \Gamma.\mathsf{encrypt}_{pk}(m')$.*

Examples of encryption schemes in the above class are ElGamal's encryption [El Gamal, 1985], the encryption scheme defined in [Boneh *et al.*, 2004a] which uses the linear Diffie-Hellman KEM, or Paillier's [Paillier, 1999] encryption scheme. In fact, these encryption schemes are homomorphic and possess an efficient protocol for proving that a ciphertext decrypts to a given plaintext: the proof of equality of two discrete logarithms [Chaum & Pedersen, 1993], in case of ElGamal or the encryption scheme in [Boneh *et al.*, 2004a], or the proof of knowledge of an N-th root in case of Paillier's encryption.

Theorem 5.9. *Let Γ be a OW-CPA secure encryption scheme from the above class $\mathbb{E}_2$. Let furthermore e be an encryption of some message under some public key* pk*. The protocol depicted in Figure 5.2 is a zero knowledge proof of knowledge of the decryption of e.*

The proof is similar to that of Theorem 4.11. □

The confirmation/denial protocol

The confirmedSign, confirmation and denial protocols of the construction in Subsection 5.2.1 are depicted in Figure 5.3.

Prover P		Verifier V
Choose $r' \xleftarrow{R} \mathbb{G}$		
Compute $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(r')$	$\xrightarrow{t_2}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{z = r' * r^b}$	
	$\xleftarrow{\mathsf{PoK}\{z = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(t_2 \circ_e e^b)\}}$	
		Accept if PoK is valid

Figure 5.2: Proof system for membership to the language $\{r : r = \Gamma.\mathsf{decrypt}(e)\}$ Common input: $(e, \Gamma.\mathsf{pk})$ and Private input: $\mathrm{r}\ \mathrm{and}\ \Gamma.\mathsf{sk}$ or randomness encrypting r in e.

Prover P		Verifier V
Compute I as defined in Definition 5.1		Compute I as defined in Definition 5.1
Choose $r' \xleftarrow{R} \mathbb{G}$		
Compute $t_1 = f(r')$		
Compute $t_2 = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(r')$	$\xrightarrow{t_1, t_2}$	
	$\xleftarrow{b}$	Choose $b \xleftarrow{R} \{0,1\}^{\ell}$ $(b \in \mathbb{N})$
	$\xrightarrow{z = r' * r^b}$	
	$\xleftarrow{\mathsf{PoK}\{z = \Gamma.\mathsf{decrypt}_{\Gamma.\mathsf{sk}}(t_2 \circ_e e^b)\}}$	
		Accept if the proof PoK is valid and, $f(z) = t_1 \circ_s I^b$ in case of confirmation, $f(z) \neq t_1 \circ_s I^b$ in case of denial.

Figure 5.3: Proof system for membership to the language $\{(e, c) : \exists r : r = \Gamma.\mathsf{decrypt}(e) \wedge c = (\neq)\Omega.\mathsf{commit}(m, r)\}$ Common input: $(e, c, m, \Gamma.\mathsf{pk}, \Omega.\mathsf{pk})$ and Private input: $\Gamma.\mathsf{sk}$ or randomness encrypting r in e.

Remark 5.3. *The prover in Figure 5.3 is either the confirmer who can run the above protocols with the knowledge of his private key, or the signer who wishes to confirm the validity of a just generated signature. In fact, with the knowledge of the randomness used to encrypt s in e, the signer can issue the above confirmation protocol thanks to the properties satisfied by Γ.*

Theorem 5.10. *The confirmation protocol (run by either the signer on a just generated signature or by the confirmer on any signature) described in Figure 5.3 is a proof of knowledge with perfect zero knowledge.*

Theorem 5.11. *The denial protocol described in Figure 5.3 is a a proof of knowledge with computational zero knowledge if the underlying encryption scheme is IND-CPA-secure.*

The proofs of both Theorem 5.10 and Theorem 5.11 are similar to those of Theorem 4.12 and Theorem 4.13 respectively □

Remark 5.4. *The protocols depicted in Figure 5.3 can be, by virtue of the result of [Damgård, 2000], efficiently turned into protocols that are ZK closed under concurrent composition in the auxiliary string model if* PoK *is a Σ protocol.*

5.3 The "signature of an encryption" paradigm

We have seen that convertible confirmer signatures realizing the "signature of a commitment" paradigm are comprised of a commitment on the message to be signed, an encryption of the random string used to produce the commitment, and a digital signature on the commitment. Since IND-CPA encryption can be easily used to get statistically binding and computationally hiding commitments, one can use instead of the commitment in the previous constructions an IND-CPA secure encryption scheme. With this choice, there will be no need to encrypt the string used to produce the encryption of the message, since the private key of the encryption scheme is sufficient to check the validity of a ciphertext w.r.t. a given message. Note that this construction already appeared in [An *et al.*, 2002] in the context of signcryption. We give below the full description of the construction.

Key generation. The signer key pair is $(\Sigma.\mathsf{pk}, \Sigma.\mathsf{sk})$ and the confirmer key pair is $(\Gamma.\mathsf{pk}, \Gamma.\mathsf{sk})$ where Σ and Γ are the digital signature and the encryption scheme underlying the construction resp.

ConfirmedSign. On input m, the signer computes an encryption $c = \Gamma.\mathsf{encrypt}_{\Gamma.\mathsf{pk}}(m)$ of m, then a digital signature $\sigma = \Sigma.\mathsf{sign}_{\Sigma.\mathsf{sk}}(c)$. Finally he outputs (c, σ) and interacts with the verifier to prove in ZK that c decrypts to obtain m. Such a proof is possible given the randomness used to encrypt m in c.

Confirmation/Denial protocol. On a message m and an alleged signature $\mu = (\mu_1, \mu_2)$, the confirmer checks the validity of μ_2 on μ_1. In case it not valid, he produces $\perp$. Otherwise, he computes the decryption $\tilde{m}$ of μ_1 and checks whether $\tilde{m} \stackrel{?}{=} m$, according to the result, he gives a ZK interactive (with the verifier) proof, using $\Gamma.\mathsf{sk}$, of the equality/inequality of the decryption of μ_1 and m.

Selective conversion. The confirmer proceeds as in the confirmation/denial protocol with the exception of issuing $\perp$ is case the signature is invalid, and a **non-interactive** proof that m is the decryption of the first field of the signature otherwise.

We notice that the construction depicted above achieves better performance than all previously cited constructions in terms of signature length, generation/verification and conversion cost. In fact, the signature contains only one encryption and a signature on it. Moreover, verification or conversion of the signature are simpler as they do not involve anymore checking whether a commitment is correctly computed. Besides, the proofs underlying the confirmedSign/confirmation/denial protocols are reduced in case of discrete-logarithm-based encryption schemes to proofs of equality/inequality of discrete logarithms for which there exists efficient protocols [Chaum & Pedersen, 1993; Camenisch & Shoup, 2003]. The only problem with this technique is the resort to non-interactive ZK (NIZK) proofs of knowledge. In fact, we know how to produce such proofs from their interactive variants using the Fiat-Shamir paradigm, which is known to provide security only in the ROM. However, the recent results in [Damgård *et al.*, 2006; Groth & Sahai, 2008; Camenisch *et al.*, 2009] exhibit efficient NIZK proofs of knowledge in some settings.

5.3.1 Security analysis

Concerning the security analysis, we first note that completeness, soundness, and the ZK property of the confirmedSign/confirmation/denial protocols are ensured by the use of ZK proofs. Next, we prove that the construction resists existential forgeries and is invisible if the underlying digital signature and encryption are SEUF-CMA and IND-CPA secure resp.

Theorem 5.12. *Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, the above construction is (t, ϵ, q_s)-EUF-CMA secure if the underlying digital signature is also (t, ϵ, q_s)-EUF-CMA secure.*

Proof. The adversary $\mathcal{R}$ against the signature underlying the construction will get the parameters of the digital signature he is trying to attack from his challenger. Then, he will choose a suitable encryption. Simulation of signatures is simple; on a query m_i, $\mathcal{R}$ computes an encryption c_i of m_i, then requests his challenger for a signature on c_i. Let σ_i be the answer of such a query. $\mathcal{R}$ will then output (c_i, σ_i) and produce a ZK proof that c_i decrypts in m_i. Such a proof, in addition to all the proofs involved in the verification/conversion queries, are possible for $\mathcal{R}$ to give with the knowledge of the encryption private key.

At some time, the adversary $\mathcal{A}$ against the construction will output a forgery $(c^\star, \sigma^\star)$ on a message $m^\star$, that was never queried before. $\sigma^\star$ is by definition a digital signature on $c^\star$. The last

item was never queried by $\mathcal{R}$ for digital signature, since otherwise $m^\star$ would have been queried before. We conclude that $(c^\star, \sigma^\star)$ is a valid forgery on the digital signature scheme. □

Theorem 5.13. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction given above is $(t, \epsilon, q_s, q_v, q_{sc})$-INV-CMA secure if it uses a (t, ϵ', q_s)-SEUF-CMA secure digital signature and a $(t + q_s(q_v + q_{sc}), \epsilon(1-\epsilon')^{q_v+q_{sc}})$-IND-CPA secure encryption scheme.*

Proof. Let $\mathcal{A}$ be the invisibility adversary against the construction, we construct an IND-CPA adversary $\mathcal{R}$ against the underlying encryption scheme as follows.

$\mathcal{R}$ gets the parameters of the target encryption scheme from his challenger, and chooses a suitable digital signature scheme. For a confirmedSign query on m_i, $\mathcal{R}$ will proceed as in the real algorithm, with the exception of maintaining a list $\mathcal{L}$ of records that consists of the query, its encryption, the randomness used to produce the encryption, and finally the digital signature on the encryption. $\mathcal{R}$ can produce digital signatures on any encryption with the knowledge of the signature scheme private key. Moreover, he can confirm any signature he has just generated with the knowledge of the randomness used in the encryption.

For a verification query (c_i, σ_i) on m_i, $\mathcal{R}$ will check $\mathcal{L}$ (after checking of course the validity of σ_i on m_i), if the record $R_i = (m_i, c_i, -, -)$ appears in the list, then he will issue a proof that c_i decrypts in m_i using the third component of the record. Otherwise, he will simulate a proof of the inequality of the decryption of c_i and m_i using the rewinding technique.
For a conversion query, $\mathcal{R}$ will proceed as in a verification query with the exception of providing the non-interactive variant of the proof he would issue if the signature is valid (using the randomness encrypting the message in the first field of the queried confirmer signature), and the symbol $\perp$ otherwise.
This simulation differs from the real one when the queried signature (c_i, σ_i) is valid on m_i however c_i does not appear in the list (as first field of the output confirmer signatures). We distinguish two cases, either the message in question m_i was not queried before for signature, in which case such a query would correspond to a valid existential forgery on the construction, and thus on the underling signature scheme. Or, the queried signature is on a message that has been queried before, which corresponds to an existential forgery on the underlying signature scheme. Since the signature scheme underlying the construction is (t, ϵ', q_s)-SEUF-CMA secure, this scenario does not happen with probability at least $(1-\epsilon')^{q_v+q_{sc}}$.

At some point, $\mathcal{A}$ produces two messages m_0, m_1. $\mathcal{R}$ will forward the same messages to his challenger and obtain a ciphertext c, encryption of m_b for some $b \xleftarrow{R} \{0,1\}$. $\mathcal{R}$ will produce a digital signature σ on c and give the result in addition to c to $\mathcal{A}$ as a challenge confirmer signature. It easy to see that $\mathcal{A}$'s answer is sufficient for $\mathcal{R}$ to conclude. Note that after the challenge phase, $\mathcal{A}$ is allowed to issue confirmedSign, verification and conversion queries and $\mathcal{R}$ can handle them as previously. There is however the possibility for $\mathcal{A}$ of issuing a verification (conversion) query of the type $(c, -) \neq (c, \sigma)$ on m_b. $\mathcal{R}$ will respond to such a query by issuing the denial protocol (symbol $\perp$). The probability that this answer does not differ from the output of the real algorithm is

at least $(1-\epsilon')^{q_v+q_{sc}}$ as the signature scheme underlying the construction is (t, ϵ', q_s)-SEUF-CMA secure by assumption. □

Remark 5.5. *Note that the IND-CPA requirement on the encryption scheme is also necessary. In fact, an invisibility adversary against the construction can easily use an IND-CPA adversary against the underlying encryption scheme in order to solve his invisibility challenge.*

5.3.2 Efficiency analysis

Confirmation/denial protocols

We showed that the confirmation (and also the confirmedSign) protocol, in confirmer signatures from the "signature of an encryption" paradigm, amounts to proving that a ciphertext encrypts a given plaintext. This is in general easy since in most encryption schemes, one can define, given a ciphertext c and its underlying plaintext m, two homomorphic one way functions f and g, and two quantities I and J such that $f(r) = I$ and $g(\mathsf{sk}) = J$, where r is the randomness used to encrypt m in c, and sk is the private key of the encryption scheme in question. Examples of such encryptions are [El Gamal, 1985], the encryption scheme defined in [Boneh *et al.*, 2004a] which uses the linear Diffie-Hellman KEM, Paillier [Paillier, 1999], and also Cramer-Shoup [Cramer & Shoup, 2003] and [Camenisch & Shoup, 2003]. The confirmation (confirmedSign) protocol in this case will be reduced to a proof of knowledge of a preimage of J (I) by the function g (f), for which we provided an efficient proof in Figure 5.1.

Concerning the denial protocol, it is not always straightforward. In most discrete-logarithm-based encryptions, this protocol amounts to a proof of inequality of discrete logarithms as in [El Gamal, 1985; Boneh *et al.*, 2004a; Cramer & Shoup, 2003]. In case the encryption scheme belongs to the class $\mathbb{E}_2$ defined in Definition 5.1, Figure 5.4 provides an efficient proof that c encrypts some $\tilde{m} \neq m$. In the protocol provided in this figure, f denotes an arbitrary *homomorphic injective one way function*:

$$f(m \star m') = f(m) \circ_s f(m')$$

With the standard tools, the above denial protocol can be shown to be a proof of knowledge with computational ZK, if the encryption scheme Γ is IND-CPA secure, and ZK closed under concurrent composition if PoK is a Σ protocol.

Selective Conversion

The selective conversion in confirmer signatures from the "signature of an encryption" paradigm consists of a non-interactive proof of the confirmation protocol. As mentioned earlier in this document, there has been recently an important progress in this area. We note in this paragraph three solutions.

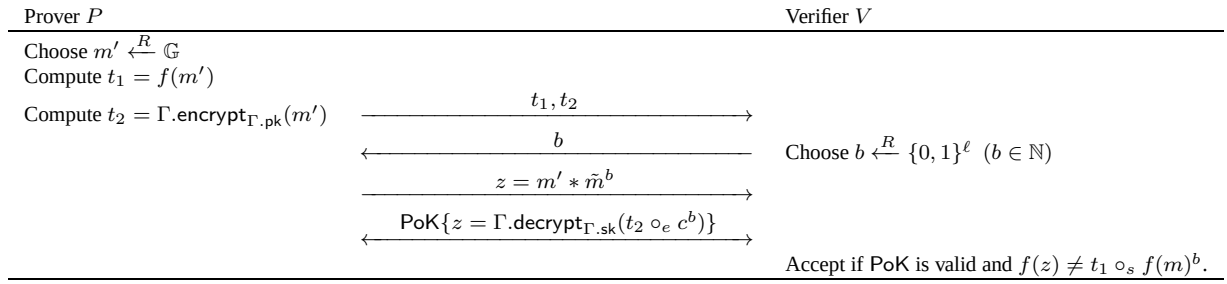

Figure 5.4: Proof system for membership to the language $\{(m, c) \colon \exists \tilde{m} : \tilde{m} = \Gamma.\mathsf{decrypt}(c) \wedge \tilde{m} \neq m\}$ Common input: $(m, c, \Gamma.\mathsf{pk})$ and Private input: $\Gamma.\mathsf{sk}$ or randomness encrypting $\tilde{m}$ in c

The case of Paillier [Paillier, 1999]'s encryption scheme. The Paillier encryption [Paillier, 1999] operates on messages in $\mathbb{Z}_N$, where $N = pq$ is a safe RSA modulus. Encryption of a message m is done by picking a random $r \in_R \mathbb{Z}_N^\times$ and then computing the ciphertext $c = r^N(1 + mN) \bmod N^2$. Decryption of a ciphertext c is first done by raising it to $\lambda = \mathrm{lcm}(p - 1, q - 1)$ to find r, then recovering m by computing $(r^{-N}c - 1)/N$. It is easy to see that Paillier's encryption belongs to what we call the class of *fully decryptable* encryption schemes, i.e. encryption schemes where decryption leads to the randomness used to produce the ciphertext. Thus, selective conversion can simply be achieved by releasing the randomness used to generate the ciphertext.

Damgård et al. [Damgård et al.*, 2006]'s solution.* This solution transforms a 3-move interactive ZK protocol P with linear answer to a non-interactive ZK one (NIZK) using a homomorphic encryption scheme in a registered key model, i.e. in a model where the verifier registers his key. This technique has been already discussed in 1.4.4 (Paragraph: non-interactive zero knowledge (NIZK)). The authors in [Damgård *et al.*, 2006] proposed an efficient illustration using Paillier's encryption and the proof of equality of two discrete logarithms. We conclude that with such a technique, the "signature of an encryption" approach accepts an efficient instantiation if the considered encryption scheme allows proving the correctness of a decryption using a proof of equality of two discrete logarithms, e.g. [El Gamal, 1985; Boneh *et al.*, 2004a; Cramer & Shoup, 2003].

Groth and Sahai [Groth & Sahai, 2008]'s solution. The authors in this work provide an efficient NIZK for the language:

$$\mathsf{PoK} = \{(a, b) \colon c_1 = u^a \wedge c_2 = v^b \wedge c_3 = g^{a+b}\}$$

The common input is $g, c_1, c_2, u, v \in (\mathbb{G}, \cdot)$ where $(\mathbb{G}, \cdot)$ is a bilinear group. The private input is either (a, b) or $(\mathsf{DL}_g(u), \mathsf{DL}_g(v))$, where $\mathsf{DL}_g(u)$ denotes the discrete logarithm of u in base g. We conclude then that the "signature of an encryption" approach accepts an efficient

instantiation if the considered encryption scheme is the one defined in [Boneh *et al.*, 2004a], since a proof of the above language can be used to prove that a given ciphertext decrypts to a given message.

5.4 Conclusion

We analyzed the security of confirmer signatures from the "signature of commitment" paradigm in the outsider security model. The plain paradigm was shown to necessitate strong encryption which makes it quite impractical, or at least allow very limited instantiations. However, a small variation results in a tremendous improvement in the efficiency. We also shed light on a particular construction which can be seen as a special sub-case of the paradigm, namely the "signature of an encryption" technique. The advantage of this technique consists in achieving better performance than the original technique (short signature, small generation, verification, and conversion cost), yet applying to any signature scheme. Its sole limitation resides in requiring efficient non-interactive proofs of knowledge. This motivates research to further tackle this problem as was started recently in [Damgård *et al.*, 2006; Groth & Sahai, 2008; Camenisch *et al.*, 2009].

Part III

Undeniable Signatures

Chapter 6

Overview of Undeniable Signatures

Abstract. Undeniable signatures, as previously mentioned in Chapter 3, were introduced in [Chaum & van Antwerpen, 1990] to limit the verification property inherent to digital signatures. In fact, the verification of undeniable signatures cannot be achieved without the cooperation with the signer. Later, this concept was upgraded to *designated confirmer signatures*, where the verification of signatures is delegated to a *designated confirmer*. Although undeniable signatures have preceded confirmer signatures by only five years, the literature on the former was so abundant that it exceeded triple the literature on the latter. In this chapter, we give a short overview of the research carried out in respect of undeniable signatures.

6.1 The genesis

Controlling the proliferation of certified copies of documents was the main motivation behind introducing undeniable signatures. In fact, it is well known that digital entities, e.g. authenticated documents, can be easily copied exactly, and as a consequence they can be subject to improper use (blackmail or industrial espionage) in case the underlying content is personally or commercially sensitive. For these reasons, Chaum and van Antwerpen introduced undeniable signatures in [Chaum & van Antwerpen, 1990] as a cryptographic primitive having all properties of digital signatures except the universal verification. In fact, the verification procedure is replaced by confirmation/denial protocols the signer issues interactively with the signature recipient.

Later in [Chaum, 1991b], Chaum polished the properties required in an undeniable signature by introducing the concept of zero-knowledgeness of the confirmation/denial protocols. In fact, after the interaction with the signer in the mentioned protocols, the signature recipient might get additional knowledge (than the signature validity/invalidity) and uses it to leak the signature status to other parties. Another attempt at refining the confirmation/denial protocols was proposed in [Fujioka *et al.*, 1991]; the authors in this work introduced the notion of interactive bi-proof systems which aim at proving concurrently which of $x \in L_1$ or $x \in L_2$ is a true theorem where L_1 and L_2

are disjoint languages. Such a notion can be employed in undeniable signatures to assure signature confirmation and disavowal with the same protocol. Chaum continued to address in [Chaum, 1991a] the potential shortcomings/misreadings (reported for instance in [Desmedt & Yung, 1991]) of his new primitive.

An important advance of undeniable signatures was suggested in [Boyar *et al.*, 1991], namely the convertibility of the undeniable signatures into publicly verifiable ones. The conversion can either be *selective*, i.e. concerns a *selected* undeniable signature, or *universal*, where the signer releases a single bit string allowing the conversion of all undeniable signatures. The authors in [Boyar *et al.*, 1991] proved the existence of convertible undeniable signatures assuming the existence of digital signatures and provided an efficient solution based on ElGamal's signature. This construction was broken and repaired in [Michels *et al.*, 1996], however the proposed scheme had only a conjectural security. Another construction of convertible undeniable signatures was given in [Damgård & Pedersen, 1996] and likewise, the security analysis was only speculative.

We finish this section by citing the works [Pedersen, 1991] and [Chaum *et al.*, 1991] which support the signer in undeniable signatures with additional features. The former allows the signer to distribute a part of his secret key to n agents such that any k of these can verify a signature, whereas the latter proposes the first undeniable signatures with unconditional security for the signer.

6.2 Combination with other primitives

The concept of undeniable signatures was so attractive that it was adopted in many other cryptographic frameworks, e.g.:

Group undeniable signatures [Lyuu & Wu, 2002]. A group signature is a cryptographic primitive which allows a member of a group to anonymously sign messages on behalf of the group. A group undeniable signature shares the same principle with group signatures with the exception of necessitating the intervention of the group manager to verify the issued signatures. This new mechanism can be for instance used to validate price lists, press releases, or digital contracts when the signatures are commercially sensitive or valuable to a competitor.

Threshold undeniable signatures [Harn & Yang, 1993; Lin *et al.*, 1996; Lee & Hwang, 1999; Wang *et al.*, 2001, 2002; Kim & Won, 2004; Guo & Tang, 2005; Chen *et al.*, 2005; Lu *et al.*, 2005]. This concept was initiated in 1992 under the name: group-oriented undeniable signatures. A group-oriented (t, n) undeniable signature scheme has the following four properties: (1) the group signature is mutually generated by at least t group members; (2) the signature verification process is simplified because there is only one group public key required; (3) the signature can only be verified with the consent of all signers; (4) the signers hold the responsibility for the signed messages. Group-oriented or threshold undeniable signatures can be for instance used in software

business where the company which vends the software is funded by a number n of investors and where the released products must be signed mutually by at least t investors.

Identity-based undeniable signatures [Libert & Quisquater, 2004; Guo & Tang, 2005; Zhang *et al.*, 2005; Wu *et al.*, 2007b; Li *et al.*, 2007]. Identity-based cryptography is a paradigm proposed by Shamir in [Shamir, 1985] to remove the necessity for public key certificates. This is achieved by letting the user's public key be an information identifying him in a non-ambiguous way (e-mail address, immatriculation number,...), and deriving the corresponding private key using the master key of a trusted authority called the private key generator (PKG). This concept was first extended to undeniable signatures in 2004 by Libert and Quisquater, and later it was applied to different primitives derived from undeniable signatures.

Undeniable multi-signatures [Yun & Lee, 2004, 2005]. An undeniable multi-signature is a signature produced by a number of signers whose cooperation is mandatory for the verification of the issued signature. Such a framework is suitable for joint copyright protection on digital contents. In fact, digital watermarks have been proposed as the means for copyright protection of multimedia data, and it is often the case that the confirmer of a watermark wants only the intended verifier to be convinced with the validity of the watermark. In case the digital multimedia content is made by co-workers, a joint copyright protection scheme is needed to provide equal right to them. Thus, undeniable multi-signatures provide a good solution in this situation.

Blind undeniable signatures [Sakurai & Yamane, 1996; Huang *et al.*, 2005; Han *et al.*, 2006; Koide *et al.*, 2008]. A blind signature enables a user to obtain a signature on a message without revealing the content of the message to the signer. Sometimes, the signer might control some attributes of the message in question such as "date of issue" or "valid until", in which case we talk about a partially blind signature. (Partially) blind signatures proved very useful in many real-life applications such as online-shopping as they protect the privacy of the user (customer) by hiding the message (purchased item) from the signer (bank). Unfortunately, the self-authenticating property of blind signatures jeopardizes completely the privacy of the signer. Thus, merging the properties of blind and undeniable signatures results in a primitive which guarantees both the privacy of the signer and of the user.

Proxy undeniable signatures [Wu *et al.*, 2007a]. A proxy signature scheme allows an entity to delegate his/her signing capability to another entity in a way that the latter can sign messages on behalf of the former when the former is not available. Proxy signatures have found numerous practical applications in ubiquitous computing, distributed systems, mobile agent applications, etc. In some situations, it is required to protect the privacy of the (proxy) signer which entails the presence of the primitive proxy undeniable signatures. In [Wu *et al.*, 2007a], the authors propose

the first convertible undeniable proxy signature scheme with rigorously proven security in the random oracle model, based on some natural complexity assumptions.

Undeniable confirmer signatures [Nguyen *et al.*, 1999]. In undeniable signatures, a signature can only be verified with the cooperation of the signer. Thus, absence of the signer obstructs the entire verification process. This problem is eliminated in confirmer signature schemes where the verification procedure is delegated to a confirmer rather than the signer. In [Nguyen *et al.*, 1999], the authors present a variation of confirmer signature, called undeniable confirmer signature in which both the signer and the confirmer can verify the validity of the signatures. Note, that such a primitive is often referred to as *directed signatures [Lim & Lee, 1993]*.

Non-interactive designated verifier undeniable signatures [Jakobsson *et al.*, 1996; Kudla & Paterson, 2005]. The seminal work of Chaum and van Antwerpen [Chaum & van Antwerpen, 1990] on undeniable signatures has been subject to many attacks. The most notable one is due to Jakobsson [Jakobsson, 1994] where he describes how the signer can be vulnerable to a black-mailing attack, i.e. a dishonest verifier can threaten the signer to broadcast the validity of a given signature if the latter does not consent to do what the former asks. Later, Jakobsson et al. [Jakobsson *et al.*, 1996] proposed a solution to this problem, called designated verifier proofs. Informally speaking, a designated verifier proof is a proof of correctness of some "statement" that either the prover or some designated verifier could have produced. If the prover created the proof, then the "statement" is correct, however a designated verifier could simulate a valid proof without a correct statement. As a result, a secure designated verifier proof will convince the designated verifier of the validity of the given statement, as he did not create the proof, but will convince no other party as the designated verifier could have generated it. Finally, it was shown in [Jakobsson *et al.*, 1996] that designated verifier proofs could be made non-interactive, however, a formal definition of non-interactive proofs of knowledge along with their applications to undeniable signatures was provided almost a decade later in [Kudla & Paterson, 2005].

6.3 RSA-based constructions

Since the introduction of undeniable signatures in 1989, a significant amount of work has been devoted to the investigation of practical schemes implementing this primitive. Up to 1997, this work was focused on discrete-log-based systems. The scheme in [Gennaro *et al.*, 2000] is the first to use regular RSA signatures to generate undeniable signatures. In this new setting, both the signature and verification exponents of RSA are kept secret by the signer, while the public key consists of a safe RSA modulus and a sample RSA signature on a single public message. The scheme possesses several attractive properties. First of all, provable security, as forging the undeniable signatures is as hard as forging regular RSA signatures. Second, both the confirmation and denial protocols

are zero-knowledge. In addition, these protocols are efficient (particularly, the confirmation protocol involves only two rounds of communication and a small number of exponentiations). Finally, the scheme in [Gennaro *et al.*, 2000] can be efficiently extended to support more advanced properties of undeniable signatures found in the literature, including convertibility of the undeniable signatures (into publicly verifiable ones), the possibility to delegate the ability to confirm and deny signatures to a third party without giving up the power to sign, and the existence of distributed (threshold) versions of the signing and confirmation operations.

Later in [Miyazaki, 2000], an improved variant of [Gennaro *et al.*, 2000], which supports the convertibility and the resilience against the hidden verifier attack (described in [Jakobsson *et al.*, 1996]), is proposed. Improvements of [Gennaro *et al.*, 2000] continued to emerge, for instance the work in [Galbraith *et al.*, 2002] proposes techniques which allow RSA-based undeniable signatures for general moduli (in contrast to the work [Gennaro *et al.*, 2000] which rests on safe RSA moduli). Additionally, the result in [Galbraith & Mao, 2003] develops an RSA-based scheme which has invisibility. Quite recently, a new approach for constructing selectively convertible RSA-based undeniable signatures without random oracles has been proposed in [Kurosawa & Takagi, 2006; Le Trieu *et al.*, 2009].

6.4 Analysis and refinement of the model

New (security) properties. The first security notions that were required in undeniable signatures were: (1) security for the verifier, which refers to the soundness of the confirmation/denial protocols, (2) unforgeability of the signatures, which refers to the hardness of producing a valid undeniable signature on an arbitrary message, (3) non-transferability and invisibility of the signatures, where non-transferability means the inability of the signature verifier to transfer his knowledge about the signature status to a third party, and invisibility connotes the difficulty of telling whether a signature is valid or not. The invisibility property had many variants; the first one requires that any polynomial adversary is incapable of distinguishing a signature based on the underlying message (the adversary outputs two messages m_0 and m_1 and receives a signature on one of those two messages; he is then required to tell the message underlying the challenge signature). There exists also the stronger notion [Galbraith & Mao, 2003] which requires the difficulty of distinguishing the signature on a message, chosen by the adversary, from a random signature in the signature space. In the same paper [Galbraith & Mao, 2003], Galbraith and Mao suggested to consider a further security property, that is anonymity, which informally means the infeasibility of determining whether a user is or is not the signer of a given message. Such a property can be the source of abuse by the signer in some situations, thus the introduction of the notion of *revocable anonymity* in [Yeung & Han, 2003; Han *et al.*, 2004] to denote the possibility of revoking the anonymity, by some trusted authority, of some signer who has done illegal actions.

Another security property that needs to be satisfied by *convertible* undeniable signatures was introduced in [Huang & Wong, 2009] and named resilience to *claimability attacks*, where a dis-

honest/malicious signer both disavows a signature via the disavowal protocol and confirms it via selective conversion. Always in the case of convertible undeniable signatures, it is desirable in some situations to delegate the ability to prove the validity and convert signatures to a semi-trusted third party by providing a verification key [Schuldt & Matsuura, 2010].
Finally, Kurosawa and Furukawa introduced in [Kurosawa & Furukawa, 2008] the notion of universal composability which informally captures the maintenance of the undeniable signature of its security properties under a general protocol composition. This notion is motivated by the fact that undeniable signatures are often used as a building block in a more complicated protocol.

Relations among security notions. The first work that addresses the relations among the different security notions of undeniable signatures is [Galbraith & Mao, 2003], where the authors prove that their notion of invisibility implies their notion of anonymity and the invisibility notion considered in [Camenisch & Michels, 2000]. They also specify some properties to be satisfied by the undeniable signature scheme in order to have invisibility in the sense of [Camenisch & Michels, 2000] and anonymity in the sense of [Galbraith & Mao, 2003] imply the strong invisibility in the sense of [Galbraith & Mao, 2003].
Besides, Kurosawa and Heng conduct in [Kurosawa & Heng, 2006] a thorough study on the unforgeability and invisibility notions of undeniable signatures in the two attack models, namely chosen message attack and full attack. In particular, they show that unforgeability against a chosen message attack (where the adversary is allowed to query adaptively the signing oracle) is equivalent to unforgeability against a full attack (where the adversary is allowed to query adaptively both the signing and the confirmation/denial oracles), and invisibility against a chosen message attack is equivalent to invisibility against a full attack.

Different types of conversion. Traditionally, the convertibility property in undeniable signatures refers to the possibility of converting an individual undeniable signature into an ordinary one (selective conversion), or publish a universal receipt that turns all undeniable signatures into publicly verifiable ones (universal conversion). Recently, convertibility in undeniable signatures has been widened to cover further features. The first example is the *time-selective conversion* property which was introduced in [Laguillaumie & Vergnaud, 2005] to circumvent the problem caused by the universal conversion of undeniable signatures. In fact, after the signer has revealed the universal trapdoor, all (*past and future*) undeniable signatures will be publicly verifiable and thus he cannot issue further undeniable signatures with his present key. As a consequence, he needs to (in case he wants to issue new undeniable signatures) generate a new key pair which has to be certified by an authority (PKI) and where the corresponding certificate needs to be generated by all the verifiers. Time-selective conversion is a notion which supports the signer to universally convert *chronologically* signatures pertaining only to a specific time period: given a time-selective convertible undeniable signature σ for a time period p, it is computationally infeasible to determine which signing secret key was used to generate σ; but with the knowledge of a matching universal receipt for some time period $p' \geq p$, it is easy to determine whether σ is a valid time-selective con-

vertible undeniable signature or not. Next, the *gradual conversion* was introduced in [El Aimani & Vergnaud, 2007] to generalize the concept of time-selective convertible undeniable signatures to *event-selective* convertible undeniable signatures where a signature becomes universally verifiable if a specific event happens and makes the signer publish the corresponding receipt information. In other words, gradual conversion enables the signer to gradually convert signatures *achronously* (*i.e.* with time periods made completely independent of each other).

6.5 Applications

The first real life application that motivated the research on undeniable signatures is the limitation of the proliferation of certified copies of a document issued by a given company. Later, Jakobsson [Jakobsson, 1994] exhibited a situation where one can use undeniable signatures for blackmailing; a malicious verifier can threaten the signer of leaking the validity of a given signature if the latter does not consent to what the former asks. This situation can be avoided if the undeniable signature scheme is well designed, namely if signatures are non-transferable.

Next, and almost a decade later, Yun and Lee provided two further applications of undeniable multi-signatures, namely the joint copyright protection on digital content [Yun & Lee, 2004] and the large scale electronic voting [Yun & Lee, 2005]. In fact, Digital watermarks have been proposed as the means for copyright protection of multimedia data. Naturally, the confirmer of a watermark wants to make sure that only the intended verifier can be convinced of the validity of the watermark and thus the need for undeniable signatures. However, existing copyright protection schemes are mainly focused on protection of single owners' copyright. In case the digital multimedia contents is made by co-workers, a joint copyright protection scheme is needed to provide equal right to them, which explains the necessity of undeniable multi-signatures. Besides, existing voting schemes assume that the voting center is trustful and untraceable channels exist between voters and the voting center. To minimize the role of the voting center, the authors in [Yun & Lee, 2005] propose a voting scheme where multiple administrators manage the voting protocol. Moreover, in the voting and counting stages, ballots cannot be opened without the help of all administrators. Also, before counting the ballot, the administrators must all verify the undeniable multi-signature on it. Finally and due to the properties of undeniable signatures, voters can change their mind to whom they vote in the registration stage. They can restart the voting process by simply rejecting the signature confirmation protocol launched by the voting manager.

The last application of undeniable signatures that has been addressed in the literature is in the area of Internet applications, or more precisely XML [Sun & Li, 2005]. XML or extensible markup language has become an important universal language for the Internet-based business world. An XML document can be generated from various resources with varying security requirements. In order to ensure the integrity of the contents in the transactions, and at the same time maintain privacy and confidentiality, security is increasingly important. The XML undeniable signatures, proposed in [Yun & Lee, 2005], are designed for the security of XML document transactions. They

guarantee the authentication, data integrity, and non-repudiation of the data they sign. Moreover, they ensure that signatures cannot be verified without interaction with the signer. The goal of a such work is to bridge the gap existing between XML technologies and data security theories in order to provide a framework for the integration of security technologies to improve XML applications.

6.6 Constructions over special algebraic structures

Popular undeniable signatures present in the literature have the disadvantage of either having long signatures, typically 1024 bits, or having operations for the signer that take cubic running time. These advantages become more tangible for some real world applications, e.g. on a chip card or a web server. Therefore, many attempts have been made to address the mentioned problems; we sketch in this section the most important such contributions.

Signatures based on ideal arithmetic in quadratic order [Biehl *et al.*, 2004]. These are signatures constructed using imaginary quadratic fields. A quadratic field is an algebraic number field of degree two over $\mathbb{Q}$. It is easy to show that the map $d \mapsto \mathbb{Q}(\sqrt{d})$ is a bijection from the set of all square-free integers $d \neq 0, 1$ to the set of all quadratic fields. If $d > 0$, the corresponding quadratic field is called a real quadratic field, and for $d < 0$, it is called an imaginary quadratic field or complex quadratic field. There has been a number of cryptographic primitives (e.g. the NICE encryption scheme [Paulus & Takagi, 2000]) using such an algebraic structure; the technique used in these systems is based on "switching" between ideals whose arithmetic is quadratic in the bit length of the public key. As a consequence, the operations on the signer's side in [Biehl *et al.*, 2004] are of quadratic complexity. The comparisons with the popular RSA-based undeniable signatures show a major advantage of [Biehl *et al.*, 2004] in terms of signature cost and length. However, the major drawback lies in the conjectural security analysis of the scheme, which becomes more improbable after the cryptanalysis of the NICE encryption scheme [Castagnos & Laguillaumie, 2009; Castagnos *et al.*, 2009].

MOVA signatures [Monnerat & Vaudenay, 2004b,a; Monnerat *et al.*, 2005]. These proposals develop a general framework based on the notion of interpolation of group homomorphisms. In this way, they define decisional and computational problems which generalize several fundamental problems found in public key cryptography, e.g. (Bilinear) Diffie-Hellman, Quadratic Residuosity, ...
These group homomorphisms allow to express well known signatures, e.g. [Chaum & van Antwerpen, 1990; Gennaro *et al.*, 2000] in a unified framework. Moreover, they allow to develop very short signatures in a quite natural way, namely by instantiating the scheme with group homomorphisms with a range group of small size. The main criticism of these signatures is the resort to the random oracle model.

Signatures using Non-Abelian groups [Thomas & Lal, 2008]. Non-Abelian groups have been considered as an alternative for doing public key cryptography. In fact, they provide a rich collection of hard problems like the *conjugacy* problem: given $x, y \in (\mathbb{G}, \cdot)$, decide whether x and y are *conjugates*, i.e. whether $\exists a \in \mathbb{G} : x = aya^{-1}$. There are many illustrations of non-Abelian groups, e.g. Braid groups, Thompson's group, Polycyclic groups. The signature presented in [Thomas & Lal, 2008] is based on the intractability of the conjugacy problem. The scheme therein does not only suffer from the conjectural security, but also from the unreasonableness of the underlying assumption; it is well known that there exists an efficient problem that solves the conjugacy problem in Braid groups.

6.7 Recent trends

We summarize in the following section the main directions of research on undeniable signatures.

Revisiting previous constructions. There have been a number of works devoted to the analysis of previous constructions of undeniable signatures. The first of such projects dates back to 2001 [Okamoto & Pointcheval, 2001] where the authors introduce a novel class of computational problems, namely the gap problems. They further show how a particular instance based on the Diffie-Hellman problems, namely the GDH problem, can serve to solve a more than 10-year old open security problem: Chaum's undeniable signature. Later, in [Ogata *et al.*, 2005], the authors improved the analysis in [Okamoto & Pointcheval, 2001], and showed that the security of the FDH variant of Chaum's scheme with NIZK confirmation and disavowal protocols is equivalent to the CDH problem. They achieve this by introducing a new kind of adversarial goal called forge-and-impersonate in undeniable signature schemes, classifying the security of the FDH variant of Chaum's undeniable signature scheme according to three dimensions, i.e. the goal of adversaries, the attacks and the ZK level of confirmation and disavowal protocols, and finally relating each security to some well-known computational problem.
The next two schemes that were revisited are those by Damgård and Pederesen [Damgård & Pedersen, 1996] and by Michels et al. [Michels *et al.*, 1996], which were addressed in [El Aimani, 2008] and [El Aimani & Vergnaud, 2007] and will be subjects of the two upcoming chapters resp. Finally, we mention the claimed attack [Li *et al.*, 2007] on Libert and Quisquater [Libert & Quisquater, 2004]'s ID-based undeniable signature; the authors show that if a valid message-signature pair has been revealed, an adversary can forge the signer's signature on any arbitrary message for which the signer has no way to deny it. This attack turns out to be flawed as the authors confuse points on an elliptic curve with elements in $\mathbb{Z}_q^{\times}$, where q is the order of the group formed by the elliptic curve points.

Generic constructions. The next direction of research was dedicated to the design of generic constructions of undeniable signatures. The first result in this line is the MOVA construction [Mon-

nerat & Vaudenay, 2004b,a; Monnerat *et al.*, 2005] described earlier in Section 6.6. Next, there is the result due to Galindo et al. [Galindo *et al.*, 2006] where the authors propose a technique for building identity based schemes with further properties. For instance, they provide a generic construction for ID-based undeniable signatures from a digital and an undeniable signature schemes. Later, the result in [Huang *et al.*, 2007a] proposes a generic construction for universally-convertible undeniable signatures; the construction is based on three building blocks: a strongly unforgeable classic signature scheme, a selectively-convertible undeniable signature scheme and a collision-resistant hash function. Finally, in [El Aimani, 2008, 2009a], we propose a generic construction of convertible undeniable signatures (both selectively and universally) from any digital signature scheme and any encryption scheme obtained from the hybrid encryption paradigm. We must also cite the construction [Zhu, 2004] which realizes the "signature of an encryption" paradigm.

Efficient signatures with strong security properties. Alleviation or removal of the idealized models and basing the security on popular and reasonable security properties was a tangible purpose in the recent proposals of undeniable signatures. We note as examples [Huang *et al.*, 2007b; El Aimani, 2008, 2009a; Le Trieu *et al.*, 2009, 2010; Schuldt & Matsuura, 2010; Huang & Wong, 2009]. It is worth noting that most of these proposals are based on the sign-then-encrypt paradigm. Moreover, efficiency, which translates in having short signatures with small generation, verification and conversion cost, was also a main intent in the recent proposals of undeniable signatures. All the previously mentioned schemes achieve also these properties as their underlying encryption layer relies on an IND-CPA secure encryption scheme. Finally, we note that it is was also desirable recently to reach a minimal number of moves between the signer and the verifier of an undeniable signature. The already mentioned signatures have constant nay four round confirmation/denial protocols. Fewer moves have been achieved by [Kurosawa & Heng, 2005; Monnerat & Vaudenay, 2005] but at the expense of security; both constructions have recourse to the random oracle model for the security analysis.

6.8 Conclusion

In this chapter, we browsed quickly through the different realizations in the area of undeniable signatures. We will continue in the next two chapters by having a closer look at two proposals, namely [Damgård & Pedersen, 1996] and [Michels *et al.*, 1996]; we will disprove the conjecture on the invisibility of the former and provide a recast of the underlying construction which achieves strong security features. Moreover, we redefine the security model of the latter so that it captures a new property, namely the *gradual conversion*, and we provide a formal security analysis of the scheme in this new model.

Chapter 7

Damgård-Pedersen's Undeniable Signatures Revisited

Abstract. Damgård-Pedersen's [Damgård & Pedersen, 1996] undeniable signatures were proposed in 1996, and consist in first generating a provably secure variant of ElGamal's signature, e.g. the Modified ElGamal signature scheme [Pointcheval & Stern, 2000], on the given message, then encrypting the message-key-dependent part using either Rabin's or ElGamal's encryption. These signatures were proven to have their unforgeability resting on the discrete logarithm problem. Concerning invisibility, it is conjectured to rest on the factorization problem in case the Rabin encryption is used, and on the DDH problem otherwise. This conjectural security was reported recently in [Kurosawa & Takagi, 2006] as the authors used a similar approach to devise their undeniable signatures.
In this chapter, we focus on the variant using ElGamal's encryption; we disprove the speculative invisibility in the model defined in [Damgård & Pedersen, 1996], and we provide a complete attack on the scheme in a very popular model. Besides, we propose a fix to the scheme which allows to achieve very strong security features; the security analysis is done in a more general framework where the refined scheme is seen as a special instantiation of this framework.
Parts of the results in this chapter appeared in the publication [El Aimani, 2009a] in the proceedings of Africacrypt 2009.

7.1 Damgård-Pedersen's undeniable signatures

7.1.1 The scheme

Let $m \in \{0,1\}^\star$ be an arbitrary message, the scheme consists of the following procedures:

Setup (setup**).** On input the security parameter κ, generate a k-bit prime t and a prime $p \equiv$

1. The prover computes s the decryption of (E_1, E_2) using ν.
Next, he chooses $s' \stackrel{R}{\leftarrow} \mathbb{Z}_t^\times$, computes and sends $t_1 = \left(g^{h(m)}h^{-r}\right)^{s'}$ and
$t_2 = (E_1\alpha^{\rho'}, s'E_2\beta^{\rho'})$ to the verifier
3. The verifier chooses $b \stackrel{R}{\leftarrow} \{0,1\}$ and sends it to the prover.
4. If $b = 0$, the prover sends s' and ρ'.
Otherwise, he sends ss' and proves that t_2 is an encryption of ss'.
5. If $b = 0$, the verifier checks that t_1 and t_2 are computed as in Step 1.
Otherwise, he checks the proof of decryption of t_2:
It it fails, he rejects the proof.
Otherwise:
If the prover is confirming the signature, the verifier accepts if $r^{ss'} = t_1$.
If the prover is denying the given signature, the verifier accepts the proof if $r^{ss'} \neq t_1$.

Figure 7.1: Proof system for membership to the language $\{(E_1, E_2, r) \in \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times \times \mathbb{Z}_p^\times \mid \exists\, s \in \mathbb{Z}_t \colon \mathsf{DL}_\alpha(\beta) = \mathsf{DL}_{E_1}(E_2 \cdot s^{-1}) \wedge g^{h(m)}h^{-r} = (\neq) r^s\}$ Common input: $(E_1, E_2, r, \mathsf{pk})$ and Private input: ν

$1 \bmod t$. Furthermore, select a collision-resistant hash function H that maps arbitrary-length messages to $\mathbb{Z}_t$.

Key generation (keygen**).** Generate g of order t, $x \in \mathbb{Z}_t^\times$, and $h = g^x \bmod p$. Furthermore, select a generator α of $\mathbb{Z}_t^\times$ and $\nu \in \{0, 1, \ldots, t-1\}$, and compute $\beta = \alpha^\nu \bmod t$. The public key is $\mathsf{pk} = (p, t, g, h, \alpha, \beta)$ and the private key is (x, ν).

Signature (sign**).** The signer first computes an ElGamal signature (s, r) on m, i.e. compute $r = g^b \bmod p$ for some $b \stackrel{R}{\leftarrow} \mathbb{Z}_t^\times$, then compute s as $h(m) = rx + bs \bmod t$. Next, he computes an ElGamal encryption $(E_1 = \alpha^\rho, E_2 = s\beta^\rho) \bmod t$, for $\rho \stackrel{R}{\leftarrow} \mathbb{Z}_{t-1}$, of s. The undeniable signature on m is the triple (E_1, E_2, r).

Confirmation/Denial protocol (confirm/deny**).** To confirm (deny) a purported signature (E_1, E_2, r) on a certain message m, the signer issues a ZKPoK of the language: (see Figure 7.1)

$$\left\{(E_1, E_2, r) \in \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times \mid \exists\, s \in \mathbb{Z}_t \colon \mathsf{DL}_\alpha(\beta) = \mathsf{DL}_{E_1}(E_2 \cdot s^{-1}) \wedge g^{h(m)}h^{-r} = (\neq) r^s\right\}$$

7.1.2 Security analysis

The above algorithms/protocols are obviously complete. Moreover, the confirmation/denial protocols are proven to be sound and zero knowledge. Finally, the signatures are proven to be unforgeable if the underlying ElGamal signature is also unforgeable, and they are conjectured (by the authors in [Damgård & Pedersen, 1996]) to meet the following security notion if the DDH problem is hard.

Definition 7.1 (Signature indistinguishability)**.** *It is defined through the following game between an attacker $\mathcal{A}$ (*a distinguisher*) and his challenger $\mathcal{R}$.*

Phase 1 *after $\mathcal{A}$ gets the public parameters of the undeniable signature scheme, namely* pk*, from $\mathcal{R}$, he starts issuing* status requests *and* signature requests*. In a status request, $\mathcal{A}$ produces a pair (m, z), and receives a 1-bit answer which is 1 iff z is a valid undeniable signature on m w.r.t.* pk*. In a signature request, $\mathcal{A}$ produces a message m and receives an undeniable signature z on it w.r.t.* pk*.*

Challenge *Once $\mathcal{A}$ decides that* ***Phase 1*** *is over, he outputs a message m and receives a string z which is either a valid undeniable signature on m (w.r.t* pk*) or a* simulated signature*, i.e. a string randomly chosen from the signature space.*

Phase 2 *$\mathcal{A}$ resumes adaptively making the previous types of queries, provided that m does not occur in any request, and that z does not occur in any status request. Eventually, $\mathcal{A}$ will output a bit.*

Let p_r, resp. p_s be the probability that $\mathcal{A}$ answers 1 in the real, resp. the simulated case. Both probabilities are taken over the random coins of both $\mathcal{A}$ and $\mathcal{R}$. We say that the signatures are indistinguishable if $|p_r - p_s|$ is a negligible function in the security parameter.

7.2 Negative Results

In this section, we provide evidence that the Damgård-Pedersen signatures are unlikely to be indistinguishable under the DDH assumption. We prove in a first stage that if there exists a *key-preserving* reduction, i.e. an algorithm launching the adversary over its own public key and other freely chosen parameters, from the DDH problem to the distinguishability of the signatures (in the sense of Definition 7.1), then there exists an efficient algorithm that solves the DDH problem. Next, we provide an actual attack on this indistinguishability in a reasonable (and popular) security model. Both attacks are based on the malleability of ElGamal's encryption; given a ciphertext, one can create another ciphertext for the same underlying message.

7.2.1 Impossibility results for key-preserving reductions

Lemma 7.1. *Assume there exists a key-preserving reduction $\mathcal{R}$ that uses an indistinguishability adversary $\mathcal{A}$ against the above scheme to solve the DDH problem. Then, there exists an efficient meta-reduction $\mathcal{M}$ that solves the DDH problem.*

As previously mentioned (Chapters 4 and 5), this lemma suggests that under the DDH assumption, there exists no key-preserving reduction from the DDH problem to the distinguishability of the signatures, and in case such an algorithm exists, then the DDH problem is easy thus rendering the reduction useless.

Proof. Let $\mathcal{R}$ be the key-preserving reduction that reduces the DDH problem to distinguishing the Damgård-Pedersen signatures in the sense of Definition 7.1. We will construct an algorithm $\mathcal{M}$ that uses $\mathcal{R}$ to solve the DDH problem by simulating a distinguisher against the signatures.

Let $(c_1 = \alpha^a, c_2 = \beta^b) \in \mathbb{Z}_t^\times \times \mathbb{Z}_t^\times$ be the DDH instance $\mathcal{M}$ is asked to solve. $\mathcal{M}$ acting as a distinguisher of the signature will make a signature request on an arbitrary message m. Let (E_1, E_2, r) be the answer to such a query. $\mathcal{M}$ will make now a status query on $(c_1 \cdot E_1, c_2 \cdot E_2, r)$ and the message m. (c_1, c_2) is a yes-Diffie-Hellman instance iff the result of the last query is the confirmation that $(c_1 \cdot E_1, c_2 \cdot E_2, r)$ is a signature on m. □

In this case, it does not seem obvious how to extend the above result to arbitrary reductions. For instance, we cannot employ the technique of non-malleability of the key generator used previously in Chapters 4 and 5. In fact, this would correspond in the current case to assume that the DDH problem, w.r.t. a given public key pk, is difficult even when given access to a CDH oracle w.r.t. any $\mathsf{pk}' \neq \mathsf{pk}$, which is untrue.

7.2.2 An attack in another security model

In Definition 7.1, the adversary or distinguisher cannot issue status signatures on the challenge message and an arbitrary signature which is different from the challenge signature. This model is very frail because it prevents the signer from issuing many signatures on the same message; once the status of a signature is known, then the status of all other signatures on the same message is also known. Thus, a more realistic model will allow the adversary to issue status queries which involve the challenge message. However, the scheme in question can be totally broken in the new setting due to the fact that, given an ElGamal ciphertext, one can create another ElGamal encryption for the same plaintext.

Lemma 7.2. *The above undeniable signatures are not indistinguishable in the presence of an adversary making status queries which comprise the challenge message.*

Proof. Let $\mathcal{A}$ be an distinguisher against the above signatures, and let (E_1, E_2, r) be the challenge signature on the challenge message m. $\mathcal{A}$ will simply choose $r \xleftarrow{R} \mathbb{Z}_{t-1}$ and make the status query on $(\alpha^r E_1, \beta^r E_2, r)$ and m. The response to such a query is sufficient for $\mathcal{A}$ to conclude as the new signature is valid on m iff the original one is also valid on m. □

7.3 Positive Results

In the previous section, we provided evidence that the Damgård-Pedersen undeniable signatures are very unlikely to be indistinguishable under the DDH assumption. This can be explained by the fact that they are not strongly unforgeable, i.e. given a signature on an arbitrary message, one can create another signature on the same message without the help of the signer. Thus, the reduction

$\mathcal{R}$ needs more than a list, maintaining the queries and their replys, in order to answer the status queries made by the distinguisher. To repair these signatures, one can first compute the ElGamal key β^ρ along with its encapsulation α^r, then produce an ElGamal signature (s, r) on the message in question concatenated with α^ρ, and finally encrypt s using β^ρ. The output undeniable signature is $(\alpha^\rho, s\beta^\rho, r)$. It is easy to see that the provided repair is a special instance of the construction in Section 4.2, and thus can be proven in a stronger security model (the resulting confirmer signatures are proven to be SINV-CMA secure) than that provided in [Damgård & Pedersen, 1996].

In the sequel, we exhibit another reduction from the anonymity of the construction to the anonymity of its underlying building blocks. In fact, although SINV-CMA security implies ANO-CMA security, however, the former rests on rather strong assumptions on the underlying building blocks, namely the IND-CPA and INV-OT security of the used KEM and DEM resp.

Theorem 7.3. *Given $(t, q_s, q_v, q_{sc}) \in \mathbb{N}^4$ and $(\varepsilon, \epsilon') \in [0,1]^2$, the construction depicted in Section 4.2 is $(t, \epsilon, q_s, q_v, q_{sc})$-ANO-CMA secure if it uses a (t, ϵ', q_s)-EUF-CMA secure, an ANO-OT secure DEM, and a $(t + q_s(q_v + q_{sc}), \frac{\epsilon}{2} \cdot (1-\epsilon')^{q_v+q_{sc}})$-ANO-CPA secure KEM.*

Proof. Let $\mathcal{A}$ be an attacker that $(t, \epsilon, q_s, q_v, q_{sc})$-ANO-CMA breaks the construction in Section 4.2, assumed to use a (t, ϵ', q_s)-EUF-CMA secure digital signature and an ANO-OT secure DEM. We will construct an algorithm $\mathcal{R}$ that $(t + q_s(q_v + q_{sc}), \frac{\epsilon}{2} \cdot (1-\epsilon')^{q_v+q_{sc}})$-ANO-CPA breaks the underlying KEM:

[Key generation] $\mathcal{R}$ gets the parameters of the KEM $\mathcal{K}$ from his challenger, namely the two public keys $\mathcal{K}.\mathsf{pk}_0$ and $\mathcal{K}.\mathsf{pk}_1$ and the encapsulation/decapsulation algorithms. Then, he chooses an appropriate ANO-OT secure DEM together with an EUF-CMA secure signature scheme Σ. He will run $\Sigma.\mathsf{keygen}$ twice to obtain $(\Sigma.\mathsf{pk}_0, \Sigma.\mathsf{sk}_0)$ and $(\Sigma.\mathsf{pk}_1, \Sigma.\mathsf{sk}_1)$. Finally he will set $\mathsf{pk}_0 = (\mathcal{K}.\mathsf{pk}_0, \Sigma.\mathsf{pk}_0)$ and $\mathsf{pk}_1 = (\mathcal{K}.\mathsf{pk}_1, \Sigma.\mathsf{pk}_1)$ as the challenge public keys for $\mathcal{A}$.

[$\mathsf{confirmedSign}$ **queries]** For a signature query on m regarding a public key pk_b, $b \in \{0,1\}$. $\mathcal{R}$ first fixes a session key k together with its encapsulation c using $\mathcal{K}.pk_b$. Then he computes a (digital) signature $\sigma = (s, r)$ on $c\|m$ using $\Sigma.\mathsf{sk}_b$. Finally, he encrypts s (using k) and outputs the result, together with r, to $\mathcal{A}$. $\mathcal{R}$ will maintain a list $\mathcal{L}_b$ of the encapsulations c and keys k used to generate the confirmer signatures with respect to the key pk_b, $b = 0, 1$.

[$\mathsf{confirm}/\mathsf{deny}$ **queries]** For a signature $\mu = (\mu_1, \mu_2, \mu_3)$ on m with respect to a given key pk_b, $b \in \{0,1\}$, $\mathcal{R}$ will look up the list $\mathcal{L}_b$. If a record having as first component the encapsulation μ_1, then $\mathcal{R}$ will use the corresponding decapsulation, say k, to decrypt (μ_1, μ_2) in s. If (s, μ_3) is a valid digital signature on $c\|m$, $\mathcal{R}$ will run the confirmation protocol, otherwise, he will run the denial protocol. $\mathcal{R}$ can issue such proofs of knowledge, without knowing the private key of $\mathcal{K}$, using the rewinding technique because the protocols are zero knowledge, thus simulatable. In case μ_1 does not appear in any record of $\mathcal{L}_b$, $\mathcal{R}$ will issue the denial protocol.

This simulation differs from the real one when the signature μ is valid and has not been obtained from a signature query. Two cases: either m was never queried to the signing oracle, then (m, μ) would correspond to an existential forgery on the confirmer signature scheme, which would lead to an existential forgery on the underlying signature scheme, by virtue of Theorem 4.8. The second case is when m has been previously queried to the signing oracle in which case (m, μ) would correspond to an existential forgery on the underlying digital scheme thanks to Remark 4.6. Hence, the probability that both scenarios do not happen is at least $(1 - \epsilon')^{q_v}$ because the underlying digital signature scheme is (t, ϵ', q_s)-EUF-CMA secure by assumption.

[convert **queries]** $\mathcal{R}$ proceeds as above with the exception of issuing the converted signature instead of the confirmation protocol, or the symbol $\perp$ instead of the denial protocol. Here, the probability that $\mathcal{A}$ does not query a valid signature that has not been obtained from a sign query is at least $(1 - \epsilon')^{q_{sc}}$.

[Challenge] Eventually, $\mathcal{A}$ outputs a challenging message $m^\star$. $\mathcal{R}$ will pick a $b' \xleftarrow{R} \{0, 1\}$ and use his challenge $(c_b^\star, k_b^\star)$ (created w.r.t. $\mathcal{K}.\mathsf{pk}_b$ for some $b \in \{0, 1\}$) to compute a digital signature $\sigma_{b'}^\star = (s_{b'}^\star, r_{b'}^\star)$, using $\Sigma.\mathsf{sk}_{b'}$, on $c_b^\star \| m^\star$. Then, he encrypts the useful part of the resulting signature ($s_{b'}^\star$) using $k_b^\star$ and outputs the result, together with $r_{b'}^\star$, as a confirmer signature $\mu^\star$ on $m^\star$. Therefore, if $b = b'$, then $\mu^\star$ is a signature on $m^\star$ with respect to pk_b, otherwise it is not a valid signature with respect to either key. If $\mathcal{A}$ has an advantage non-negligibly different from that of an adversary in a real attack (as described in Definition 3.4), then $\mathcal{A}$ can be used to used to break the ANO-OT security of the DEM; actually $r_{b'}^\star$ reveals by assumption no information about $\Sigma.\mathsf{pk}_{b'}$.

[Post challenge phase] $\mathcal{A}$ will continue issuing queries to the signing, confirmation/denial, and selective conversion oracles, with respect to the two keys pk_0 or pk_1, and $\mathcal{R}$ can answer as previously. Note that in this phase, $\mathcal{A}$ might request the verification or selective conversion of a confirmer signature $(c_b^\star, -, -)$ on a message m_i with respect to pk_b, $b = 0, 1$. In this case, $\mathcal{R}$ will simply issue the denial protocol in case of a verification query, or the symbol $\perp$ in case of a selective conversion query. Following the same analysis above, the probability that the simulation does not differ from the real execution is at least $(1 - \epsilon')^{q_{sc}+q_v}$.

[Final output] When $\mathcal{A}$ outputs his answer $b_a \in \{0, 1\}$, $\mathcal{R}$ will output $b'' = b'$ to his challenger in case $b_a = b'$, and $b'' = 1 - b'$ otherwise. We clearly have $\epsilon = |\Pr[b_a = b' | b = b'] - \frac{1}{2}|$. The advantage of $\mathcal{R}$ is defined by:

$$
\begin{aligned}
\mathsf{Adv}(\mathcal{R}) &= (1-\epsilon')^{q_v+q_{sc}} \left| \Pr[b = b''] - \frac{1}{2} \right| \\
&= (1-\epsilon')^{q_v+q_{sc}} \left| \Pr[b_a = b', b = b''] + \Pr[b_a \neq b', b = b''] - \frac{1}{2} \right| \\
&= (1-\epsilon')^{q_v+q_{sc}} \left| \Pr[b_a = b', b = b'] + \Pr[b_a \neq b', b \neq b'] - \frac{1}{2} \right| \\
&= (1-\epsilon')^{q_v+q_{sc}} \left| \Pr[b_a = b' | b = b'] \Pr[b = b'] + \Pr[b_a \neq b' | b \neq b'] \Pr[b \neq b'] - \frac{1}{2} \right| \\
&= \frac{(1-\epsilon')^{q_v+q_{sc}}}{2} \left| \Pr[b_a = b' | b = b'] - \frac{1}{2} \right| \\
&= \frac{\epsilon(1-\epsilon')^{q_v+q_{sc}}}{2}
\end{aligned}
$$

The last but one equation is due to the fact that $\Pr[b = b'] = \frac{1}{2}$ as $b' \xleftarrow{R} \{0,1\}$, and to that fact that $\Pr[b_a \neq b' | b \neq b'] = \frac{1}{2}$ since the used DEM is ANO-OT secure.

□

7.4 Conclusion

In this chapter, we revisited the Damgård-Pedersen [Damgård & Pedersen, 1996] undeniable signatures which had a conjectural security left open for over a decade. We disproved the invisibility of these signatures in the model given in [Damgård & Pedersen, 1996], and provided a complete attack in a stronger model which is quite reasonable. Next, we proposed a fix to these signatures so that they become invisible; interestingly, this repair turns out to be a special instantiation of the construction provided in Section 4.2. Actually, even the confirmation/denial protocols provided in [Damgård & Pedersen, 1996] happen to be a special case of the confirmation/denial protocols provided for the construction in Section 4.2. Moreover, we provided another analysis of the construction in question which establishes its anonymity based on the anonymity of its components. We conclude that the construction in Section 4.2 does not only capture the efficient realizations of confirmer/undeniable signatures proposed recently, e.g. [Le Trieu *et al.*, 2010; Schuldt & Matsuura, 2010], but also serves for analyzing the early schemes that have a speculative security.

Chapter 8

Gradually Convertible Undeniable Signatures

Abstract. In 1990, Boyar, Chaum, Damgård, and Pedersen introduced in [Boyar *et al.*, 1991] *convertible undeniable signatures* which limit the self-authenticating property of digital signatures but can be converted by the signer to ordinary signatures. Six years later, Michels, Petersen, and Horster presented in [Michels *et al.*, 1996] an attack on the El Gamal-based seminal scheme of Boyar *et al.*, and proposed a repaired version without formal security analysis. In this chapter, we modify their scheme so that it becomes a generic one, and it provides an advanced feature which permits the signer to universally convert *achronously* all signatures pertaining to a specific time period. We supply a formal security treatment of the modified scheme: we prove, in the generic group model, that the scheme is existentially unforgeable and invisible under chosen message attacks, assuming reasonable assumptions on the underlying constituents.
Parts of the results in this chapter appeared in the joint work [El Aimani & Vergnaud, 2007] with Damien Vergnaud in the proceedings of ACNS 2007.

8.1 Gradually convertible undeniable signatures

8.1.1 Syntax

Let $\pi \in \mathbb{N}$. A *gradually convertible undeniable signature scheme* US with π time periods consists of the following procedures:

Setup ($\mathsf{US.setup}$**).** This is an algorithm which takes an integer k as input, and outputs the *public parameters* $\mathsf{Parameters}$. κ is called the *security parameter*.

Signer key generation ($\mathsf{US.skeygen}$**).** This algorithm takes the public parameters as input and

outputs a pair $(\mathbf{sk_s}, \mathbf{pk_s})$, where $\mathbf{sk_s}$ is called the *signing private key* and $\mathbf{pk_s}$ the *signing public key*.

Verifier key generation ($\mathsf{US.vkeygen}$**).** This algorithm inputs the public parameters and outputs a pair $(\mathbf{sk_v}, \mathbf{pk_v})$, where $\mathbf{sk_v}$ is called the *verifying private key* and $\mathbf{pk_v}$ the *verifying public key*.

Signature ($\mathsf{US.sign}$**).** This algorithm takes the public parameters, a message, an integer in $[\![1, \pi]\!]$, and a signing private key as inputs and outputs a bit string.

Verification ($\mathsf{US.verify}$**).** This algorithm, run by the signer, inputs the public parameters, a message m, a bit string μ, an integer $p \in [\![1, \pi]\!]$, and a signing key pair $(\mathbf{sk_s}, \mathbf{pk_s})$ and outputs a bit which is equal to 1 iff the bit string μ is a valid undeniable signature on m for the time period p w.r.t. $\mathbf{pk_s}$.

Confirmation/Denial protocols ($\mathsf{US.\{confirm, deny\}}$**).** These are two-party protocols (P, V) between the signer P and a signature recipient V such that:

- P and V take as common input a message m, an integer $p \in [\![1, \pi]\!]$, a bit-string μ, a signing public key $\mathbf{pk_s}$, a verifying public key $\mathbf{pk_v}$, and the public parameters;
- P takes as private input $\mathbf{sk_s}$ the signing secret key corresponding to $\mathbf{pk_s}$;
- V takes as private input $\mathbf{sk_v}$ the verifying secret key corresponding to $\mathbf{pk_v}$;
- (P, V) is a proof of the validity/invalidity of the purported signature μ on the message m for the time period p w.r.t. the public key $\mathbf{pk_s}$.

At the end of the protocols, the verifier V either accepts or rejects the proof.

Selective conversion ($\mathsf{US.convert}$**).** This is an algorithm which takes as input the public parameters, an integer in $[\![1, \pi]\!]$, a signing key pair and a bit string Υ (either a pair message/signature or the empty string) and outputs a bit string.

Selective verification ($\mathsf{verifyConverted}$**).** This is an algorithm that takes as input the public parameters, a message m, a bit string μ, an integer $p \in [\![1, \pi]\!]$, a signing public key $\mathbf{pk_s}$, and a bit string Λ and outputs a bit. If the bit output is 1 then the bit string Λ is said to be a *receipt* of the validity of μ.

8.1.2 Security model

Standard properties

Let π be an integer. For all $\kappa \in \mathbb{N}$, for all $\mathsf{Parameters} \in \mathsf{US.setup}[\kappa]$, for all $(\mathbf{pk_s}, \mathbf{sk_s}) \in \mathsf{US.skeygen}[\mathsf{Parameters}]$, for all $m \in \{0,1\}^*$ and for all $p \in [\![1, \pi]\!]$:

1. The protocols US.confirm and US.deny are designated verifier proofs of membership for the languages (respectively):

$$\{(\mathsf{Parameters}, m, \mu, p, \mathbf{pk_s}) \big| \mathsf{US.verify}[\mathsf{Parameters}, m, \mu, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}\}$$
$$\{(\mathsf{Parameters}, m, \mu, p, \mathbf{pk_s}) \big| \mathsf{US.verify}[\mathsf{Parameters}, m, \mu, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{0\}\}$$

   where $(\mathsf{Parameters}, m, \mu, p, \mathbf{pk_s}) \in \mathsf{US.setup}[k] \times \{0,1\}^{*^2} \times [\![1, \pi]\!] \times \mathsf{US.skeygen}[\mathsf{Parameters}]$.

2. $\forall \mu \in \mathsf{US.sign}[\mathsf{Parameters}, m, p, \mathbf{sk_s}]$:

$$\mathsf{US.verify}[\mathsf{Parameters}, m, \mu, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}.$$

3. $\forall \mu \in \mathsf{US.sign}[\mathsf{Parameters}, m, p, \mathbf{sk_s}], \forall \Lambda \in \mathsf{US.convert}[\mathsf{Parameters}, p, (\mathbf{sk_s}, \mathbf{pk_s}), (m, \mu)]$:

$$\mathsf{US.verifyConverted}[\mathsf{Parameters}, m, \mu, p, \mathbf{pk_s}, \Lambda] = \{1\}$$

4. $\forall \mu, \Lambda \in \{0,1\}^*$:

   $\mathsf{US.verifyConverted}[\mathsf{Parameters}, m, \mu, p, \mathbf{pk_s}, \Lambda] = \{1\} \Rightarrow \mathsf{US.verify}[\mathsf{Parameters}, m, \mu, p, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}$.

The first property captures the validity and the non-transferable property of the protocols confirm and deny (*i.e.* the use of designated verifier proofs insures that a verifier will gain no information in an execution of one of these protocols [Kudla & Paterson, 2005]). The last three properties are the properties of *correctness*:

- a well-formed signature is always accepted by the algorithm verify;
- a receipt correctly constructed is always accepted by the algorithm verifyConverted;
- and if there exists a bit-string Λ which makes accepted a bit-string μ by the algorithm verifyConverted, then μ is a valid signature.

Existential unforgeability

As previously mentioned, the standard notion of security for digital signatures was defined in [Goldwasser *et al.*, 1988] as *existential unforgeability against adaptive chosen message attacks* (EUF-CMA). In [Laguillaumie & Vergnaud, 2005], the corresponding notion for time-selective convertible undeniable signatures is defined along the same lines. The definition of *resistance to forgery* for gradually convertible undeniable signatures that we propose is similar. In fact, we suppose that the adversary has access to the universal receipts for every time period $p \in [\![1, \pi]\!]$ and is allowed to query a signing oracle $\mathfrak{S}$ for any message of its choice. As usual, in the adversary's answer, there is the natural restriction that the returned message/signature has not been obtained from the signing oracle.

Definition 8.1 (Unforgeability - EUF-CMA)**.** *Let π be a positive integer, let* $\mathsf{US} = (\mathsf{setup}, \mathsf{skeygen}, \mathsf{vkeygen}, \mathsf{sign}, \mathsf{verify}, \mathsf{confirm}, \mathsf{deny}, \mathsf{convert}, \mathsf{verifyConverted})$ *be a gradually convertible undeniable signature scheme with π time periods and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, where κ is a security parameter:*

Experiment $\mathbf{Exp}^{\mathsf{euf-cma}}_{\mathsf{US},\mathcal{A}}(\kappa)$

$\mathsf{Parameters} \xleftarrow{R} \mathsf{US.setup}(\kappa)$,
$(\mathbf{pk_s}, \mathbf{sk_s}) \xleftarrow{R} \mathsf{US.skeygen}(\mathsf{Parameters})$
$(\mathbf{pk_v}, \mathbf{sk_v}) \xleftarrow{R} \mathsf{US.vkeygen}(\mathsf{Parameters})$
for j *from* 1 *to* π *do*
  $\Lambda_j \leftarrow \mathsf{US.convert}(\mathsf{Parameters}, j, \mathbf{pk_s}, \mathbf{sk_s}, \varepsilon)$
$(m^\star, \mu^\star, p^\star) \leftarrow \mathcal{A}^{\mathfrak{S}}(\mathsf{Parameters}, \mathbf{pk_s}, \mathbf{pk_v}, \mathbf{sk_v}, \{\Lambda_j\}_{j\in[\![1,\pi]\!]})$
  $\mid\ \mathfrak{S} : (m, p) \longmapsto \mathsf{US.sign}(\mathsf{Parameters}, m, p, \mathbf{sk_s})$
return 1 if and only if the following properties are satisfied:
  - $\mathsf{US.verify}[\mathsf{Parameters}, m^\star, \mu^\star, p^\star, (\mathbf{sk_s}, \mathbf{pk_s})] = \{1\}$
  - *m was not queried to* $\mathfrak{S}$

We define the success *of $\mathcal{A}$* via*:*

$$\mathbf{Succ}^{\textit{euf-cma}}_{\mathsf{US},\mathcal{A}}(k) = \Pr\left[\mathbf{Exp}^{\textit{euf-cma}}_{\mathsf{US},\mathcal{A}}(k) = 1\right].$$

Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0, 1]$, $\mathcal{A}$ is called a (t, ε, q_s)-EUF-CMA adversary against US *if, running in time t and issuing q_s signing queries, $\mathcal{A}$ has $\mathbf{Succ}^{\textit{euf-cma}}_{\mathsf{US},\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme* US *is said to be (t, ε, q_s)-EUF-CMA secure if no (t, ε, q_s)-EUF-CMA adversary against it exists. Finally, we consider an undeniable signature scheme* US *with security parameter $\kappa \in \mathbb{N}$,* $\mathsf{US}(\kappa)$ *is said to be EUF-CMA secure if, for any polynomial functions $t, q_s : \mathbb{N} \to \mathbb{N}$ and any non-negligible function $\varepsilon : \mathbb{N} \to [0, 1]$, it is $(t(\kappa), \varepsilon(\kappa), q_s(\kappa))$-EUF-CMA secure.*

Remark 8.1. *Note that the adversary in the above definition is not given the confirmation/denial and selective conversion oracles. In fact, these oracles are useless for him as he has the universal receipts $\{\Lambda_j\}_{j\in[\![1,\pi]\!]}$ at his disposal.*

Invisibility

We state the precise definition of *invisibility* under a chosen message attack (INV-CMA) which captures the notion that an attacker cannot distinguish signatures based on their underlying messages. We consider an INV-CMA-adversary $\mathcal{A}$ that runs in two stages. In the find stage, it takes as input a signing public key $\mathbf{pk_s}$ and outputs two different messages $m_0^\star$ and $m_1^\star$, and a time period $p^\star$ together with some state information $\mathcal{I}$. In the guess stage, $\mathcal{A}$ gets a challenge gradually convertible undeniable signature $\mu^\star$ formed by signing at random one of the challenge messages for the time period $p^\star$ under $\mathbf{pk_s}$, and it must say which message was signed. In both stages, the adversary has access to a signing oracle $\mathfrak{S}$ for $\mathbf{pk_s}$. The attacker is also given the universal receipts of the signer for all[1] time period $p \in [\![1, \pi]\!] \setminus \{p^\star\}$. The only restriction on $\mathcal{A}$ is that $p^\star$ should not arise, as a time period, in any signature request.

[1]This is the main difference with time-selective convertible undeniable signatures from [Laguillaumie & Vergnaud, 2005] where these universal receipts were given only for $p \in [\![1, p^\star - 1]\!]$.

Definition 8.2 (Invisibility - INV-CMA)**.** *Let π be a positive integer, let* $\mathsf{US} = (\mathsf{setup}, \mathsf{skeygen}, \mathsf{vkeygen}, \mathsf{sign}, \mathsf{control}, \mathsf{confirm}, \mathsf{deny}, \mathsf{convert}, \mathsf{verify})$ *be a gradually convertible undeniable signature scheme with π time periods and let $\mathcal{A}$ be a PPTM. We consider the following random experiment, for $b \in \{0,1\}$, where k is a security parameter and $b \xleftarrow{R} \{0,1\}$:*

Experiment $\mathbf{Exp}^{\mathsf{inv\text{-}cma}-b}_{\mathsf{US},\mathcal{A}}(\kappa)$

$\mathsf{Parameters} \xleftarrow{R} \mathsf{US}.\mathit{Setup}(\kappa)$

$(\mathbf{pk_s}, \mathbf{sk_s}) \xleftarrow{R} \mathsf{US}.\mathit{sKeyGen}(\mathsf{Parameters})$,

$(\mathbf{pk_v}, \mathbf{sk_v}) \xleftarrow{R} \mathsf{US}.\mathsf{vkeygen}(\mathsf{Parameters})$

$(m_0^\star, m_1^\star, p^\star, \mathcal{I}) \xleftarrow{R} \mathcal{A}^{\mathfrak{S}}(\mathit{find}, \mathsf{Parameters}, \mathbf{pk_s}_0, \mathbf{pk_s}_1)$

$\quad \mid \ \mathfrak{S} : (m, p \in [\![1,\pi]\!] \setminus \{p^\star\}) \longmapsto \mathsf{US}.\mathsf{sign}(\mathsf{Parameters}, m, p, \mathbf{sk_s})$

$\mu^\star \leftarrow \mathsf{US}.\mathsf{sign}(\mathsf{Parameters}, m_b^\star, p^\star, \mathbf{sk_s})$

for j *from* 1 *to* π *do*

$\quad \Lambda_j \leftarrow \mathsf{US}.\mathsf{convert}(\mathsf{Parameters}, j, \mathbf{pk_s}, \mathbf{sk_s}, \varepsilon)$

$d \leftarrow \mathcal{A}^{\mathfrak{S},\mathfrak{C}\mathfrak{v},\mathfrak{V}}(\mathit{guess}, \mathcal{I}, \{\Lambda_j\}_{j \in [\![1,\pi]\!] \setminus \{p^\star\}})$

Return d

We define the advantage $\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{US},\mathcal{A}}(\kappa)$ *of* $\mathcal{A}$ via*:*

$$\left| \Pr\left[\mathbf{Exp}^{\mathsf{inv-cma-b}}_{\mathsf{US},\mathcal{A}}(\kappa) = b\right] - \frac{1}{2} \right|.$$

Given $(t, q_s) \in \mathbb{N}^2$ and $\varepsilon \in [0,1]$, $\mathcal{A}$ is called a (t, ε, q_s)-INV-CMA adversary against US *if, running in time t and issuing q_s signing queries, $\mathcal{A}$ has $\mathbf{Adv}^{\mathsf{inv-cma}}_{\mathsf{US},\mathcal{A}}(\kappa) \geq \varepsilon$. The scheme* US *is said to be (t, ε, q_s)-INV-CMA secure if no (t, ε, q_s)-INV-CMA adversary against it exists. Finally, we consider an undeniable signature scheme* US *with security parameter $\kappa \in \mathbb{N}$;* $\mathsf{US}(\kappa)$ *is said to be INV-CMA secure if, for any any polynomial functions $t, q_s : \mathbb{N} \to \mathbb{N}$, and any non-negligible function $\varepsilon : \mathbb{N} \to [0,1]$, it is $(t(\kappa), \varepsilon(\kappa), q_s(\kappa))$-INV-CMA secure.*

Remark 8.2. *Note that the adversary in the above definition is not given the confirmation/denial and selective conversion oracles. In fact, these oracles are useless for him as he has the universal receipts $\{\Lambda_j\}_{j \in [\![1,\pi]\!]} \setminus \{p^\star\}$ at his disposal.*

8.2 Hash functions and new security properties

Hash functions, as previously mentioned in this document, take messages of arbitrary length and output a fixed length string. In cryptographic uses of a hash function $\mathcal{H} : \{0,1\}^* \longrightarrow H$, these properties are considered prerequisite:

- *Preimage resistance*: given $h \in H$, it should be computationally intractable to find a message m such that $\mathcal{H}(m) = h$.
- *Collision-resistant:* it should be computationally intractable to find two different messages m_1 and m_2 such that $\mathcal{H}(m_1) = \mathcal{H}(m_2)$.

In this section, we formulate the generalization of these security notions and study their properties.

8.2.1 Definitions

The security proof of our variant of Michels-Petersen-Horster's signatures makes use of new non-standard variations of the preimage resistance and the collision resistance assumptions for hash functions. These assumptions are of independent interest as they have interesting relations with the classical ones. We call them *random affine preimage resistance* and *random linear collision resistance*. Although stronger than the standard assumptions, they are quite realistic.

According to [Rogaway & Shrimpton, 2004], a hash function family is a family of functions $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$, where $\mathcal{K}_k$ is a finite non-empty set. We will write the first argument of $\mathcal{H}_k$ as a subscript, so that $\mathcal{H}_{K,k}(m) = \mathcal{H}_k(K,m)$. In the following, we denote elements from $\{0,1\}^k$ as the corresponding k-bits integers in binary representation and we will denote for every integer $N \in \mathbb{Z}$, $\mathcal{H}^N_{K,k}$ the map defined by: $\mathcal{H}^N_{K,k} : \left\{ \begin{array}{ccc} \{0,1\}^* & \longrightarrow & \mathbb{Z}_N \\ m & \longmapsto & \mathcal{H}_{K,k}(m) \mod N. \end{array} \right.$

The new security definitions can be quantified as follows:

Definition 8.3 (Random affine preimage resistance)**.** *Let n be an integer, let $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$ be a hash function family, and let $\mathcal{A}$ be a PPTM. The success $\mathbf{Succ}^{raPre(n)}_{\mathcal{H},\mathcal{A}}(k)$ of $\mathcal{A}$ against the n-random affine preimage resistance of $\mathcal{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ is defined by:*

$$\max_{\substack{K\in\mathcal{K}_k \\ 2^{k-1}\leq N<2^k \\ \alpha_1,\ldots,\alpha_n\in\mathbb{Z}^*_N \\ \beta_1,\ldots,\beta_n\in\mathbb{Z}^*_N}} \left\{ \Pr \left[\begin{array}{c} K \xleftarrow{R} \mathcal{K}_k \\ (m,i,j) \leftarrow \mathcal{A}(K,\alpha_1,\ldots,\alpha_n,\beta_1,\ldots,\beta_n) \\ m \in \{0,1\}^*, (i,j) \in [\![1,n]\!]^2, i \neq j \\ \alpha_i + \beta_j \mathcal{H}^N_{K,k}(m) = 0 \mod N \end{array} \right] \right\}.$$

An adversary $\mathcal{A}$ against the n-random affine preimage resistance of a hash function family $(\mathcal{H}_k)_{k\in\mathbb{N}}$ can be transformed easily into an adversary against the classical preimage resistance of $(\mathcal{H}_k)_{k\in\mathbb{N}}$ with success probability greater than $\mathbf{Succ}^{raPre(n)}_{\mathcal{H},\mathcal{A}}(k)/n^2$ and time-complexity of $\mathcal{A}$ increased by the time necessary to compute n modular multiplications modulo N. In particular, the 1-random affine preimage resistance is equivalent to the classical preimage resistance.

Definition 8.4 (Random linear collision resistance)**.** *Let n be an integer, let $(\mathcal{H}_k : \mathcal{K}_k \times \{0,1\}^* \longrightarrow \{0,1\}^k)_{k\in\mathbb{N}}$ be a hash function family and let $\mathcal{A}$ be a PPTM. The success $\mathbf{Succ}^{rlColl(n)}_{\mathcal{H},\mathcal{A}}(k)$ of $\mathcal{A}$ against the n-random affine preimage resistance of $\mathcal{H} = (\mathcal{H}_k)_{k\in\mathbb{N}}$ is defined by:*

$$\max_{\substack{K\in\mathcal{K}_k \\ 2^{k-1}\leq N<2^k \\ \lambda_1,\ldots,\lambda_n\in\mathbb{Z}^*_N}} \left\{ \Pr \left[\begin{array}{c} K \xleftarrow{R} \mathcal{K}_k; (m,m',i,j) \leftarrow \mathcal{A}(K,\lambda_1,\ldots,\lambda_n) \\ m,m' \in \{0,1\}^*, (i,j) \in [\![1,n]\!]^2, m \neq m' \\ \lambda_i \cdot \mathcal{H}_{K,N}(m) = \lambda_j \cdot \mathcal{H}_{K,N}(m') \mod N \end{array} \right] \right\}.$$

As for random affine preimage resistance, the 1-random linear collision resistance is equivalent to the classical collision resistance. Unfortunately, the n-random linear collision resistance cannot be reduced generically to the collision resistance for $n \geq 2$.

Remark 8.3. *This security requirement is however reasonable since if the hash function family underlying the protocol RSA-FDH [Bellare & Rogaway, 1993] does not satisfy it, then it is existential forgeable against a* one *chosen-message attack: given an RSA public key* (N, e)*, the adversary can simply pick at random* $r_1, \ldots, r_n \in \mathbb{Z}_N$*, compute* $\lambda_i = r_i^e \mod N$ *for all* $i \in [\![1, n]\!]$*, and try to find a random linear collision with parameters* $N, \lambda_1, \ldots, \lambda_n$*. If a collision* $m, m' \in \{0,1\}^*, (i, j) \in [\![1, n]\!]^2$ *(such that* $\lambda_i \cdot \mathcal{H}_{K,N}(m) = \lambda_j \cdot \mathcal{H}_{K,N}(m') \mod N$*) is found, then the adversary queries the signature* σ *on* m *to the signing oracle and can compute the signature of* m' *as* $\sigma' = r_i \cdot \sigma \cdot r_j^{-1} \mod N$.

8.2.2 Generic security

The best known general collision-finding attack against a hash function family is the so-called birthday-attack. If we assume that the values of the hash-function family $(\mathcal{H}_k)_{k\in\mathbb{N}}$ are uniformly distributed over $\{0,1\}^k$ and that the generalization of the birthday attack[2] against the random affine preimage resistance and the random linear collision resistance of $(\mathcal{H}_k)_{k\in\mathbb{N}}$ is the best possible attack (which is true in the random oracle model), then it is possible to give exponential lower bounds on the minimum of n and of the number of hash function evaluations required to have non-negligible probability of success. Indeed, for any integer $N \geq 2$, and for $(i, k) \in \mathbb{Z}_N$, it is straightforward [Stadje, 2002] that:

$$\#\{j \in \mathbb{Z}_N | i \cdot j \text{ rem } N \leq k\} = \gcd(i, N) \times \left(\left\lfloor \frac{k}{\gcd(i, N)} \right\rfloor + 1 \right).$$

Therefore if D denotes the product of two independent random variables uniformly distributed over $\mathbb{Z}_N$, we have $\forall k \in \mathbb{Z}_N$

$$\Pr(D \leq k) = \frac{1}{N^2} \sum_{i=0}^{N-1} \gcd(i, N) \left(\left\lfloor \frac{k}{\gcd(i, N)} \right\rfloor + 1 \right),$$

and consequently, D is close to the uniform distribution over $\mathbb{Z}_N$. The results from [Bellare & Kohno, 2004] are sufficient to conclude.

[2]These attacks consist in picking messages $m_1, \ldots, m_r$, computing $h_i = \mathcal{H}_k(m_i) \mod N$ for $i \in [\![1, r]\!]$ and $\gamma_{i,j} = -h_i\beta_j \mod N$ (*resp.* $\gamma_{i,j} = h_i\lambda_j \mod N$) for $j \in [\![1, n]\!]$. They are successful if there is a triple $(i, j, \ell) \in [\![1, r]\!] \times [\![1, n]\!]^2$ (*resp.* a 4-tuple $(i, i', j, j') \in [\![1, r]\!]^2 \times [\![1, n]\!]^2$) s. t. $\gamma_{i,j} = \alpha_\ell$ (*resp.* $\gamma_{i,j} = \gamma_{i',j'}$ and $j \neq j'$).

8.3 Michels-Petersen-Horster's convertible undeniable signatures revisited

8.3.1 Description of the scheme

Let π be an integer. We describe in this section our variant of Michels-Petersen-Horster's scheme. It is parameterized by a prime order group generator [Bellare *et al.*, 2001], a hash function family, and two pseudo-random function families [Rogaway & Shrimpton, 2004].

Let $\mathbb{G}$ be a group of prime order q. A *reduction function* is a map that sends an element of the group $\mathbb{G}$ [Brown, 2005; Stern *et al.*, 2002] to an integer in $\mathbb{Z}_q$. In our security analysis, the reduction function must satisfy the so called *almost-invertibility*: given an arbitrary integer in $\mathbb{Z}_q$, then, with non-negligible probability, one can efficiently find one preimage.

Definition 8.5. *Let F be a reduction function $F : \mathbb{G} \to \mathbb{Z}_q$. An almost-inverse of F is a probabilistic algorithm G, possibly outputting $\perp$, such that:*

$$\Pr_{b \xleftarrow{R} \mathbb{Z}_q}[G(b) \in \mathbb{G} \wedge F(G(b)) = b] \geq \frac{1}{3}.$$

A reduction function F is (δ, t)-almost-invertible with almost-inverse G if furthermore no distinguisher, running in time t, between $\mathcal{D} = \{G(b) \mid b \xleftarrow{R} \mathbb{Z}_q \wedge G(b) \in \mathbb{G}\}$ and $\mathcal{U} = \{a \mid a \xleftarrow{R} \mathbb{G}\}$ can get an advantage greater than δ.

The scheme US

Setup (US.setup**):** on input a security parameter κ, output a group $\mathbb{G}$ of prime order q generated by an element P, a reduction function $F : \mathbb{G} \to \mathbb{Z}_q$, a hash function $h : \{0,1\}^* \to \mathbb{Z}_q$, and two pseudo-random functions $H^1 : \mathbb{Z}_q \times [\![1, \pi]\!] \to \{0,1\}^\kappa$ and $H^2 : \{0,1\}^\kappa \times \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_q$. The public parameters are $(q, \mathbb{G}, P, h, H^1, H^2)$.

Signer key generation (US.skeygen**):** the signer picks at random its secret key $u, v \xleftarrow{R} [\![1, q-1]\!]$, computes $U \leftarrow uP$ and $V \leftarrow vP$, and sets (U, V) as its public key.

Verifier key generation (US.vkeygen**):** the verifier picks at random its secret key $w \xleftarrow{R} [\![1, q-1]\!]$, computes $W \leftarrow wP$, and sets it as its public key.

Signature (US.sign**):** on message m and period p, the signer does the following:

- $r \xleftarrow{R} [\![1, q-1]\!]$, $R \leftarrow rP$.
- $e_p \leftarrow H^1_v(p)$, $d \leftarrow H^2_{e_p}(m, R)$, $T \leftarrow dP$. If $F(T) = 0$, it tries with another value r.
- $s \leftarrow (F(T) \cdot d \cdot h(m) \cdot v - u \cdot F(R) - 1) r^{-1} \bmod q$.

The signature is the tuple (R, T, s).

Verification ($\mathsf{US.verify}$**):** to check the validity of a signature (R, T, s), the signer checks, using his private key v, that:

$$(v \cdot F(T) \cdot h(m))T = F(R)U + sR + P. \tag{8.1}$$

Confirmation/Denial protocols ($\mathsf{US.\{confirm, deny\}}$**):** the signer provides a designated verifier proof of the equality/inequality of two discrete logarithms, namely $F(R)U + sR + P$ to the base $(F(T).h(m))T$ and V to the base P (see Section 8.3.2).

Selective conversion ($\mathsf{US.convert}$**):** there exist two types of conversions, namely:

- The gradual conversion of signatures corresponding to the time period p could be done by releasing the value e_p.
- The individual conversion can be achieved by releasing the value of d.

Selective verification ($\mathsf{verifyConverted}$**):** the signature corresponding to the period p, once e_p or d is revealed, could be checked by any verifier using the equations: $(d \cdot F(T) \cdot h(m))V = F(R)U + sR + P$ and $T = dP$.

8.3.2 Proofs of equality/inequality of discrete logarithms

Let $\mathbb{G}$ be a group with prime order q. To confirm or deny that a bit string is a signature in our undeniable signature scheme, it is necessary to prove that a given quadruple $(U_1, V_1, U_2, V_2) \in \mathbb{G}^4$ is a Diffie-Hellman quadruple (or not), *i.e.* belongs to the set $\mathsf{EDL}(\mathbb{G}) = \{(x, U_1, V_1, U_2, V_2) \in \mathbb{Z}_q^\times \times \mathbb{G}^4, x = \mathsf{DL}_{U_1}(V_1) = \mathsf{DL}_{U_2}(V_2)\}$ (or to the set $\mathsf{IDL}(\mathbb{G}) = \mathbb{G}^4 \setminus \mathsf{EDL}(\mathbb{G})$). In our case, x, U_1, U_2, V_1, V_2 correspond to d, P, $F(T) \cdot h(m)V$, T, and $F(R)U + sR + P$ respectively.

To face *blackmailing* or *mafia* attacks against our undeniable signatures, we use interactive designated verifier proofs, as introduced in [Jakobsson *et al.*, 1996] by Jakobsson, Sako, and Impagliazzo, in Chaum's proofs of equality (*cf.* Fig. 8.1) and inequality (*cf.* Fig. 8.2) of discrete logarithm of [Camenisch & Shoup, 2003]. The idea is to replace the generic commitment scheme by a *trapdoor commitment* [Jakobsson *et al.*, 1996] and using classical techniques, the proofs are readily seen to be complete, sound, and above all non-transferable. The protocols involve a point $Y = yU_1$, where y is the secret key of the verifier, and the prover must be convinced that Y is well-formed (in the registered public key model, the registration procedure is used to force the users to know the secret-key corresponding to their public key).

Protocol EDL.Prove
Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input: x
$\mathcal{V}$'s output: δ
① $\mathcal{P} \xrightarrow{C_1, C_2, C_3} \mathcal{V}$
$(a, b, k) \xleftarrow{R} [\![1, q-1]\!]^3$
$C_1 \leftarrow kU_1$; $C_2 \leftarrow kU_2$
$C_3 \leftarrow aU_1 + bY$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c} \mathcal{V}$
$c \leftarrow k - x(r+b) \mod q$
• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow cU_1 + (r+b)V_1$
$\widetilde{C_2} \leftarrow cU_2 + (r+b)V_2$
$\widetilde{C_3} \leftarrow aU_1 + bY$
if $(C_1, C_2, C_3) = (\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3})$
then $\delta \leftarrow$ Accept **else** $\delta \leftarrow \perp$

Protocol EDL.Fake
Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input: y
$\mathcal{V}$'s output: δ
① $\mathcal{P} \xrightarrow{C_1, C_2, C_3} \mathcal{V}$
$(c, d, k) \xleftarrow{R} [\![1, q-1]\!]^3$
$C_1 \leftarrow cU_1 + dV_1$; $C_2 \leftarrow cU_2 + dV_2$
$C_3 \leftarrow kU_1$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c} \mathcal{V}$
$b \leftarrow d - r \mod q$; $a \leftarrow k - by \mod q$
• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow cU_1 + (r+b)V_1$
$\widetilde{C_2} \leftarrow cU_2 + (r+b)V_2$
$\widetilde{C_3} \leftarrow aU_1 + bY$
if $(C_1, C_2, C_3) = (\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3})$
then $\delta \leftarrow$ Accept **else** $\delta \leftarrow \perp$

Figure 8.1: Interactive designated verifier proof of membership of the language $\mathsf{EDL}(\mathbb{G})$

Protocol IDL.Prove
Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input : x
$\mathcal{V}$'s output : δ
① $\mathcal{P} \xrightarrow{C_0, C_1, C_2, C_3} \mathcal{V}$
$(a, b, k_0, k_1, k_2) \xleftarrow{R} [\![1, q-1]\!]^5$
$C_0 \leftarrow k_0(V_2 - xU_2)$
$C_1 \leftarrow k_1 U_1 - k_2 V_1$
$C_2 \leftarrow k_1 U_2 - k_2 V_2$
$C_3 \leftarrow aU_1 + bY$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c, d} \mathcal{V}$
$c \leftarrow k_1 - xk_0(r+b) \mod q$
$d \leftarrow k_2 - k_0(r+b) \mod q$
• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow cU_1 - dV_1$
$\widetilde{C_2} \leftarrow C_0 + cU_2 - (r+b)V_2$
$\widetilde{C_3} \leftarrow aU_1 + bY$
if $(C_1, C_2, C_3) = (\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3}) \wedge C_0 \neq \mathbb{O}_{\mathbb{G}_2}$
then $\delta \leftarrow$ Accept **else** $\delta \leftarrow \perp$

Protocol IDL.Fake
Common input: (U_1, U_2, V_1, V_2), Y
$\mathcal{P}$'s input: y
$\mathcal{V}$'s output: δ
① $\mathcal{P} \xrightarrow{C_0, C_1, C_2, C_3} \mathcal{V}$
$(c, d, k_1, k_2) \xleftarrow{R} [\![1, q-1]\!]^4$
$C_0 \xleftarrow{R} \mathbb{G} \setminus \{\mathbb{O}_{\mathbb{G}}\}$; $C_1 \leftarrow cU_1 - dV_1$
$C_2 \leftarrow C_0 + cU_2 - k_1 V_2$
$C_3 \leftarrow k_2 U_1$
❶ $\mathcal{V} \xrightarrow{r} \mathcal{P}$
$r \xleftarrow{R} [\![1, q-1]\!]$
② $\mathcal{P} \xrightarrow{a, b, c, d} \mathcal{V}$
$b \leftarrow k_1 - r \mod q$; $a \leftarrow b - k_2 y \mod q$
• $\mathcal{V}$'s execution ending
$\widetilde{C_1} \leftarrow cU_1 - dV_1$
$\widetilde{C_2} \leftarrow C_0 + cU_2 - (r+b)V_2$
$\widetilde{C_3} \leftarrow aU_1 + ybY$
if $(C_1, C_2, C_3) = (\widetilde{C_1}, \widetilde{C_2}, \widetilde{C_3}) \wedge C_0 \neq \mathbb{O}_{\mathbb{G}_2}$
then $\delta \leftarrow$ Accept **else** $\delta \leftarrow \perp$

Figure 8.2: Interactive designated verifier proof of membership to the language $\mathsf{IDL}(\mathbb{G})$

8.4 Security analysis

We first note that the property of non-transferability is fulfilled by our scheme as a direct consequence of the use of designated-verifier proofs in the confirm/deny protocols. Further, we state that our scheme resists existential forgeries and that signatures are invisible. Both security reductions stand in the generic group model [Shoup, 1997].

8.4.1 The generic group model

As mentioned in Subsection 1.3.4, a generic group infers the presence of "encodings" of the group elements instead of explicit formulas. More specifically, given an additive group $\mathbb{G}$ with prime order q and non-identity element P, one can define a map $\sigma\colon \mathbb{Z}_q \to S \subset \{0,1\}^\star$ such that the bit-string $\sigma(i)$, $i \in \mathbb{Z}_q$, represents the group element iP.

A generic algorithm $\mathcal{A}$ will then consult the group $\mathbb{G}$'s oracle for queries of type $(\overrightarrow{i}, \overrightarrow{\alpha})$, where $\overrightarrow{i}$ refers to the set of considered group elements given by their encodings $\sigma(i)$, $i \in \overrightarrow{i}$ ($\mathcal{A}$ does not know necessarily the i's), whereas $\overrightarrow{\alpha}$ denotes the set of exponents. The oracle will re-

spond to such a query with a randomly selected bit-string representing the encoding of the element $(\sum_{\alpha_i \in \overrightarrow{\alpha}, i \in \overrightarrow{i}} (i\alpha_i))P$.

We can give an interpretation of the oracle's behavior regarding such a type of queries using polynomials over $\mathbb{F}_q$. In fact, let $\mathcal{L} = [z_0, z_1, z_2, z_3, \ldots, z_{n+3}]$ be the sequence of queries' answers where n denotes the total number of queries to the group oracle. We use an interpretation similar to that in [Stern *et al.*, 2002]; the oracle will maintain, in addition to the outputs list $\mathcal{L}$, a further list of polynomials $F_i(X, Y)$ over $\mathbb{F}_q$, which we denote by $\mathcal{F}$. The lists are updated as follows:

- Polynomials F_0, F_1, F_2, F_3 are set to $F_0 = 0$, $F_1 = 1$, $F_2 = X$ and $F_3 = Y$ which correspond to the neutral element $\mathbb{O}_{\mathbb{G}}$, the generator P, and the public keys U and V respectively. The corresponding bit-strings are z_0, z_1, z_2, z_3 respectively.
- At the ℓ-th query $(\overrightarrow{i}, \overrightarrow{\alpha})$, the polynomial F_ℓ is defined as $\sum_{j=1}^{|\overrightarrow{\alpha}|} \overrightarrow{\alpha}_j \mathcal{F}_{\overrightarrow{i}_j}$. If F_ℓ is already listed as F_h, then F_ℓ is *marked* and the corresponding answer to F_h is returned. Otherwise, z_ℓ is selected at random from S, recorded together with its corresponding polynomial F_ℓ in $\mathcal{L}$ and $\mathcal{F}$ respectively and then returned to $\mathcal{A}$.

It is easy to see that the simulation driven by this interpretation is similar to that of the regular algorithm provided that all answers corresponding to the new polynomials are distinct and that no non-zero polynomial $F_i - F_j$, where i and j range the $n + 4$ polynomials indices in $\mathcal{F}$, vanishes at $(X, Y) = (u, v)$. In these conditions, we call the sequence of encodings $\mathcal{L}$ a safe sequence. We measure the probability of such a sequence using the following lemmas [Stern *et al.*, 2002]:

Lemma 8.1 (Schwartz-Zippel)**.** *Let P be a non-zero affine bivariate polynomial in $\mathbb{F}_q[X, Y]$, then:*

$$\Pr_{x,y \in F_q}[P(x, y) = 0] \leq 1/q.$$

□

Lemma 8.2. *If $n^2 \leq q$ then the probability of unsafe sequences is upper-bounded by $(n + 4)^2/q$.*

Proof. The proof is similar to [Stern *et al.*, 2002], however, we exhibit it since our generic model is slightly different.
We first note that the probability that the sequence of encodings $\mathcal{L}$ is constituted by distinct bit-strings z_i's (corresponding to new queried polynomials F_i) is exactly $\prod_{i=1}^{n+3}(1 - \frac{i}{q})$. Thus the probability that the z_i's are not all distinct is:

$$1 - \prod_{i=1}^{n+3}(1 - \frac{i}{q}) \leq 1 - (1 - \sum_{i=1}^{n+3} \frac{i}{q}) \leq \frac{(n + 3)(n + 4)}{2q}.$$

Now, once the list $\mathcal{L}$ is set, we use Lemma 8.1 to bound the probability that, among the queried polynomials F_i, there exist non-identical polynomials F_i and F_j evaluating to the same value at

the point (u, v), or equivalently, there exists a non-zero polynomial $F_i - F_j$ vanishing at (u, v). Since there are at most $\binom{n+4}{2}$ possible polynomials, such a probability is upper-bounded by $\binom{n+4}{2}/q = (n+3)(n+4)/2q$.
Summing up the two probabilities, we get the announced result. □

Remark 8.4. *All the polynomials F_ℓ are affine bivariate, i.e. of the form $aX + bY + c$. Moreover, in case one of the private keys u or v is revealed, for instance v in the universal conversion, the polynomials F_ℓ in play become affine univariate (of the form $aX + b$ where the indeterminate X refers to the public key U).*

Finally, a security proof in this model assures the absence of an adversary who behaves generically with respect to the given group. However, a security proof in the generic model does not rule out the existence of a successful adversary for a specific group [Dent, 2002; Stern *et al.*, 2002].

8.4.2 Resistance to forgery

The theorem below states that our variant of Michels-Petersen-Horster's scheme is EUF-CMA-secure in the generic group model assuming the preimage resistance, the random affine preimage resistance and the random linear collision resistance of the underlying hash function family.

Theorem 8.3. *Let $\mathcal{A}$ be an EUF-CMA-adversary in the generic group model, operating in time t, after n group queries and m signing queries, such that $m \ll n^2$ and $n \gg 1$, with success probability $\mathbf{Succ}^{\text{euf-cma}}_{\mathsf{US},\mathcal{A}}$.*

There exist adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ operating in time t' against the n-random affine preimage resistance, the n-random linear collision resistance, and the preimage resistance of the underlying hash function (respectively) such that:

$$t' \leq t + 5n\tau_G \ln n + m(\tau_{H^1} + \tau_h + 5 \ln n(\tau_G + \tau_{H^2}) + \tau_F)$$

and

$$5 \cdot \mathbf{Succ}^{raPre(n)}_{h,\mathcal{B}} + \mathbf{Succ}^{rlColl(n)}_{h,\mathcal{C}} + 6 \cdot n^2 \mathbf{Succ}^{Pre(n)}_{h,\mathcal{D}} \geq \frac{\mathbf{Succ}^{\text{euf-cma}}_{\mathsf{US},\mathcal{A}}}{9} - 12n^4/q - 6mn^2\delta_G - 12mn^3/q$$

where δ_G is the advantage of an adversary playing a distinguisher for G, and τ_G, τ_F, τ_{H^1}, τ_{H^2} and τ_h are the running times for G, F, H^1, H^2, and h respectively.

The EUF-CMA-adversary $\mathcal{A}$ will output a valid signature $\sigma^\star = (R^\star, T^\star, s^\star)$ on a message $m^\star$ for the time period $p^\star$ with success probability $\mathbf{Succ}^{\text{euf-cma}}_{\mathsf{US},\mathcal{A}}$. In our security analysis, this event is divided into sub-events according to whether $R^\star$ or $T^\star$ were created during the simulation by a signature query or by a group query.

In the lists used to maintain the group oracle, a group element created during a group query will have a "group" tag, while the tags "signR" and "signT" will correspond to elements created in a signature query. In fact, a signature query on a message m for the time period p will be answered by a triple (R, T, s), where $R, T \in \mathbb{G}$, hence the need to specify whether the element was created as an R or a T.

Remark 8.5. *The procedure that adds a new element z_ℓ to the list will be denoted* `Record`$(z_\ell \| F_\ell \| t_l)$*, where F_ℓ and t_ℓ are the corresponding polynomial and tag respectively.*

The different forgeries output by $\mathcal{A}$ will be classified as follows:

- **Type 0:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$,
- **Type 1:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{signR}$,
- **Type 2:** $\mathsf{Tag}(R^\star) = \mathsf{group}$, $\mathsf{Tag}(T^\star) = \mathsf{signT}$,
- **Type 3:** $\mathsf{Tag}(R^\star) = \mathsf{signR}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$,
- **Type 4:** $\mathsf{Tag}(R^\star) = \mathsf{signR}$, $\mathsf{Tag}(T^\star) = \mathsf{signR}$,
- **Type 5:** $\mathsf{Tag}(R^\star) = \mathsf{signR}$, $\mathsf{Tag}(T^\star) = \mathsf{signT}$,
- **Type 6:** $\mathsf{Tag}(R^\star) = \mathsf{signT}$, $\mathsf{Tag}(T^\star) = \mathsf{group}$,
- **Type 7:** $\mathsf{Tag}(R^\star) = \mathsf{signT}$, $\mathsf{Tag}(T^\star) = \mathsf{signR}$,
- **Type 8:** $\mathsf{Tag}(R^\star) = \mathsf{signT}$, $\mathsf{Tag}(T^\star) = \mathsf{signT}$.

We denote ε_θ the probability that the forgery $\sigma^\star = (R^\star, T^\star, s^\star)$ output by $\mathcal{A}$ is of type **Type** θ (for $\theta \in \{0, \ldots, 8\}$). We have:

$$\sum_{\theta=0}^{8} \varepsilon_\theta = \mathbf{Succ}^{\mathsf{euf\text{-}cma}}_{\mathsf{US},\mathcal{A}}$$

The adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ against the n-random affine preimage resistance, the n-random linear collision resistance, and the preimage resistance of the underlying hash function respectively will simulate the group and signing oracles according to the alleged kind of forgery returned by $\mathcal{A}$. More precisely, adversary $\mathcal{C}$ will use the forgery to find a random linear collision if it is of type **Type 5**, $\mathcal{D}$ will exploit a forgery of type **Type 0** to break the preimage resistance and finally, the adversary $\mathcal{B}$ will utilize all the remaining cases to find a random affine preimage.
Finally, in our unforgeability proof, we assume that $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ have revealed the private key v (universal conversion) so that $\mathcal{A}$ is able to check the validity of the answers to his signature queries. It follows that the confirmation/denial oracles are useless for him. Also, the adversaries $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ will manipulate affine univariate polynomials during the group oracle simulation, i.e. they will receive queries of type (a, b) corresponding to the polynomial $aX + b$.

Remark 8.6. *The reduction $\mathcal{R}$ (anyone of the adversaries $\mathcal{B}, \mathcal{C}$ or $\mathcal{D}$) will force $\mathcal{A}$ to return a tuple $(R^\star, T^\star, s^\star)$ such that $R^\star = 0_{\mathbb{G}}$ when the forgery is of **Type 4** and $F(T^\star) = 0$ when it is of **Type 8**. Therefore, the reduction must guess correctly when the forgery is of the given type, then simulate the group and signing oracles accordingly. In these cases, The adversary $\mathcal{A}$ will fail to return a valid forgery, thus $\epsilon_4 = \epsilon_8 = 0$, granted that the reduction doesn't abort, i.e. provides a perfect simulation of the group/signing oracles. We will denote the probability that the reduction fails in the above cases by $\Pr[\mathcal{R} \text{ aborts}]$. Theses latter quantities will be deduced from the overall success of $\mathcal{R}$ according to the following elementary lemma:*

$$\Pr[A \wedge \neg B] \geq \Pr[A|\neg B] - \Pr[B]$$

Proof. Let $(R^\star, T^\star, s^\star)$ be the forgery output by $\mathcal{A}$ on the message $m^\star$ for the time period $p^\star$. Due to the similarities in the reduction's behavior, we will detail only the case where the forgery is of type **Type 2** and give a sketch of the other cases.

Description of $\mathcal{B}$. $\mathcal{B}$ picks uniformly at random an integer $\theta \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ which is his guess for the type of the forgery output by $\mathcal{A}$. In the following simulation, we suppose that $\mathcal{A}$ returns a forgery of type **Type 2** and that $\theta = 2$ ($\mathcal{B}$ has correctly guessed the forgery's type).

The forgery produced by A satisfies the following equation[3] :

$$a - b \cdot F(R^\star) = (ad - bc) \cdot v \cdot F(T^\star) \cdot h(m),$$

where $R^\star = aU + bP$ and $T^\star = cU + dP$. Since $T^\star$ was generated during a signature query as a "T" ($\mathsf{Tag}(T) = \mathsf{signT}$), we have $c = 0$ (the verification of the signature involves the verification of equation 8.1 and of $T = dP$). Hence, the equation turns out to be $a - b \cdot F(R) = a \cdot d \cdot v \cdot F(T) \cdot h(m)$ or

$$1 - \frac{b}{a} \cdot F(R) = d \cdot v \cdot F(T) \cdot h(m).$$

Thus, in order to find a random affine preimage, $\mathcal{B}$ must plug the values α and β in answers to the group and to the signature queries (respectively). More precisely, he must answer group queries (a, b) by R such that $1 - \frac{b}{a} \cdot F(R) = \alpha$. Similarly, signature queries must be answered by (R, T, s), such that $-d \cdot v \cdot F(T) = \beta$:

Game 0. We consider an EUF-CMA-adversary $\mathcal{A}$ in the generic group model. In any game **Game i**, we denote S_i the event "$(R^\star, T^\star, s^\star)$ is a valid forgery of type **Type 2** and $\theta = 2$". By definition, we have $\Pr[S_0] = \varepsilon_2/9$.

[3] this follows from the verification equation 8.1.

Game 1. We use the interpretation described above for the generic oracle which considers a safe sequence $\mathcal{L}$. This game differs from the previous one only on unsafe sequences. Using the Lemma 8.2 we get:

$$|\Pr[S_1] - \Pr[S_0]| \leq (n+4)^2/q.$$

Game 2. In this game we modify the simulation of the group oracle. On query (a_i, b_i) such that the corresponding polynomial $F_i(X) = a_i X + b_i$ is new, $\mathcal{B}$ does the following:

- $\mathrm{ctr} \leftarrow 0$[4]
- `Repeat`
  `Pick` the next α_i in the instance of the random affine preimage problem raPre(n);
  `Compute` $r_i \leftarrow (1-\alpha_i)a_i b_i^{-1}$;
  `Compute` $\tilde{R}_i \leftarrow G(r_i)$;
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$;
  `Until` $(\tilde{R}_i \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$;
- `Pick` $R_i \xleftarrow{R} S$;
- `Return` R_i;

The event S_2 differs from the previous one if $\tilde{R}_i$ remains undefined. Since the experiments are mutually independent (a_i and b_i are uniformly distributed), we may use a lemma from elementary probability theory [Stern *et al.*, 2002, Lemma 5] to bound the corresponding probability by $1/n^2$. The overall probability when i ranges the set of queries indices is then $1/n$. Hence, we have:

$$\Pr[S_2] \geq (1 - 1/n) \Pr[S_1].$$

Game 3. In this simulation, the group oracle replaces R_i from the previous game by $\tilde{R}_i$. It executes - `Record`$(R_i \| a_i X + b_i \| \mathsf{group})$ and returns the new value of R_i as a response to the oracle query. Since the inputs to G are uniformly distributed (α_i is picked at random), we can use n times the almost-invertibility of F (the so-called *hybrid technique*) to bound the probability of S_3:

$$|\Pr[S_3] - \Pr[S_2]| \leq n\delta_G.$$

Game 4. In this game, $\mathcal{B}$ simulates the signing oracle. On query (m_j, p_j) it does the following:

- `Compute` $e_p \leftarrow H_v^1(p_j)$;
- `Pick` $R_j \xleftarrow{R} S$;
- `Compute` $d_j \leftarrow H_{e_p}^2(m_j, R_j)$;
- $\mathrm{ctr} \leftarrow 0$;

[4] In the remaining of the chapter, ctr denotes a counter ranging from 0 to $5 \ln n$.

- `Repeat`
  `Pick` the next β_j in the instance of raPre(n);
  `Compute` $t_j \leftarrow -d_j^{-1} v^{-1} \beta_j$;
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$;
  `Until` $(G(t_j) \neq \mathsf{Fail}) \cup (ctr = 5 \ln n)$;
- $T_j \leftarrow G(t_j)$;
- `Pick` $a_j \xleftarrow{R} \mathbb{Z}_q^*$;
- `Compute` $s_j \leftarrow -F(R_j) \cdot a_j^{-1}$;
- `Return` (R_j, T_j, s_j);

This game differs from the previous one if $G(t_j)$ remains undefined, which occurs with probability less that m/n^2, or if T_j is distinguished from a uniformly random string in S, in which case the probability is upper-bounded by $m\delta_G$ (using the same hybrid technique). Thus:

$$\Pr[S_4] \geq (1 - m/n^2) \Pr[S_3] - m\delta_G.$$

Game 5. In this game, $\mathcal{B}$ adds the element T to the list, maintained by the group oracle, using the command $\texttt{Record}(T_j \| d_j \| \mathsf{signT})$. This game differs from the previous one if leads to inconsistencies in the simulation of the group oracle, namely when d_j (as a polynomial) collides with another polynomial in $\mathcal{F}$. Since R_j was drawn uniformly at random from S, and d_j is value of $H^2_{e_p}$ at (m_j, R_j), the probability of having such a collision is upper-bounded by n/q:

$$|\Pr[S_5] - \Pr[S_4]| \leq mn/q.$$

Game 6. In this game $\mathcal{B}$ computes $b_j \leftarrow a_j(\beta_j h(m_j) - 1)F(R_j)^{-1}$ and adds R_j, together with its corresponding polynomial $F_j(X) = a_j X + b_j$ to the lists maintained by the group oracle by executing the command $\texttt{Record}(R_j \| a_j X + b_j \| \mathsf{signR})$. Again, due to the randomness of a_j, the difference between the previous game is:

$$|\Pr[S_6] - \Pr[S_5]| \leq mn/q.$$

Game 7. In this game, $\mathcal{B}$ exploits the forgery $(R^\star, T^\star, s^\star)$ returned by $\mathcal{A}$. We have supposed that $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{group}, \mathsf{signT})$ and $\mathcal{B}$ generated the correct θ, thus, there exist i, j such that $R^\star = R_i, T^\star = T_j$ and $1 - \frac{a_i}{b_i} F(R_i) = \alpha_i$ and $-d_j \cdot v \cdot F(T_j) = \beta_j$. The equation satisfied by the forgery turns out to be $\alpha_i + \beta_j h(m) = 0$. $\mathcal{B}$ would then find a random affine preimage with success probability:

$$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq (1 - m/n^2)(1 - 1/n)(\epsilon_2/9 - (n+4)^2/q) - (n - m/n + m)\delta_G - 2mn/q$$

Procedure:

① In signature queries (m_j, p_j): create R_j such that $b_j = 0$.
② Reject the forgery since $R^\star = \mathbb{O}_\mathbb{G}$ (see 8.3.1)

Group queries (a_i, b_i)**:** Do as in 8.4.1.

Signature queries (m_j, p_j)**:**

- $\texttt{Compute}\ e_p \leftarrow H_v^1(p_j)$ • $\mathrm{ctr} \leftarrow 0$
- `Repeat`:
  $\texttt{pick}\ R_j \xleftarrow{R} S$
  $\texttt{compute}\ d_j \leftarrow H_{e_p}^2(m_j, R_j)$
  $\texttt{compute}\ t_j \leftarrow d_j^{-1} v^{-1} h(m_j)^{-1}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$
  $\texttt{Until}\ (G(t_j) \neq \perp) \cup (\mathrm{ctr} = 5 \ln n)$
- $\texttt{Pick}\ a_j \xleftarrow{R} \mathbb{Z}_q$
- $\texttt{Record}(R_j \| a_j X \| \mathsf{signR})$ • $\texttt{Record}(T_j \| d_j \| \mathsf{signT})$
- $\texttt{Compute}\ s_j \leftarrow F(R_j) a_j^{-1} \bmod q$
- $\texttt{Return}\ (R_j, T_j, s_j)$

Final output:

$a_j^1 - b_j^1 F(R_j^1) = (a_j^1 b_j^2 - a_j^2 b_j^1) F(R_j^2) \cdot v \cdot h(m^\star)$.
or $a_j^1 = b_j^1 = 0$ thus $R_j^1 = 0_\mathbb{G}$. The forgery is then rejected.

Advantage and time (ϵ' and t') of $\mathcal{B}$:

$\Pr[\mathcal{R} \text{ aborts}] \leq n^2/q + m\delta_G + 2mn/q$
and
$t_4 \leq t + n + m(\tau_{H^1} + 5 \ln n(\tau_G + \tau_{H^2}) + \tau_h + \tau_F)$

(a) **Type 4:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{signR}, \mathsf{signR})$

Procedure:

① Reject the forgery since $F(T_j) = 0$ (see 8.3.1)

Group queries (a_i, b_i)**:** Do as in 8.4.1.

Signature queries (m_j, p_j)**:**

- $\texttt{Pick}\ R_j \xleftarrow{R} S$
- $\texttt{Compute}\ e_p \leftarrow H_v^1(p_j)$
- $\texttt{Compute}\ d_j \leftarrow H_{e_p}^2(m_j, R_j)$
- $\mathrm{ctr} \leftarrow 0$
- `Repeat`: $\texttt{pick}\ a_j, b_j \xleftarrow{R} \mathbb{Z}_q$
  $\texttt{compute}\ t_j \leftarrow (a_j - b_j \cdot F(R_j)) a_j^{-1} d_j^{-1} v^{-1} h(m_j)^{-1}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$
  $\texttt{Until}\ (G(t_j) \neq \perp) \cup (\mathrm{ctr} = 5 \ln n)$
- $\texttt{Record}(R_j \| a_j X + b_j \| \mathsf{signR})$
- $\texttt{Record}(T_j \| d_j \| \mathsf{signT})$
- $\texttt{Compute}\ s_j \leftarrow F(R_j) a_j^{-1} \bmod q$
- $\texttt{Return}\ (R_j, T_j, s_j)$

Final output:

$-d_j F(T_j) = 0$ thus $F(T_j) = 0$, so reject the forgery.

Advantage and time of $\mathcal{B}$:

$\Pr[\mathcal{R} \text{ aborts}] \leq n^2/q + m\delta_G + 2mn/q$
and
$t_8 \leq t + n + m(\tau_{H^1} + \tau_{H^2} + \tau_h + 5\tau_G \ln n + \tau_F)$

(b) **Type 8:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{signT}, \mathsf{signT})$

or[5]

$$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq \epsilon_2/9 - n^2/q - (n+m)\delta_G - 2mn/q$$

and time

$$t_2 \leq t + 5n \ln n + m(\tau_{H^1} + \tau_{H^2} + 5\tau_G \ln n + \tau_h + \tau_F).$$

We refer to Appendix for the treatment of forgeries of of **Type 1, 3, 4, 6, 7, 8**. We provide in Figures 8.3(a), 8.3(b),8.3(c),8.3(d),8.3(e), and 8.3(f) the behavior of $\mathcal{B}$ when processing the forgeries of **Type 4, 8, 1, 3, 6, 7** resp. We will consider that to the group query (a_i, b_i), $\mathcal{B}$ will respond with R_i, and to the signature query on (m_j, p_j), he will answer (R_j, T_j, s_j), where $R_j = a_j U + b_j$ and $T_j = c_j U + d_j$.

Description of $\mathcal{C}$. $\mathcal{C}$ will provide a simulation which exploits a forgery $(R^\star, T^\star, s^\star)$ of the type **Type 5**. Hence $\mathcal{C}$ will simulate the group oracle in the standard way described in 8.4.1. Furthermore, it will plug the λ_j's (instance of the random linear collision problem) in answers to signature queries such that the returned signature (R_j, T_j, s_j) satisfies $d_j \cdot v \cdot F(T_j) = \lambda_j$. In this way, the returned forgery $(R^\star, T^\star, s^\star) = (R_i, T_j, s^\star)$ will satisfy the following:

[5] In the proofs that follow, we consider that $m \ll n^2$ and $n \gg 1$, in order to simplify the expressions.

Procedure:

① In group queries (a_i, b_i):
- plug α_i in $\frac{a_i}{b_i} - F(R_i)$.

② In signature queries (m_j, p_j):
- create R_j such that $b_j = 0$,
- plug β_j in $a_j \cdot v \cdot F(R_j)$.

Group queries (a_i, b_i)**:**

- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` the next α_i in the raPre(n) instance
  `compute` $r_i \leftarrow (\frac{a_i}{b_i} - \alpha_i)$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(r_i) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(R_i = G(r_i) \| a_i X + b_i \| \mathsf{group})$
- `Return`(R_i)

Signature queries (m_j, p_j)**:**

- `Compute` $e_p \leftarrow H_v^1(p_j)$
- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` $R_j \xleftarrow{R} S$
  `compute` $d_j \leftarrow H_{e_p}^2(m_j, R_j)$
  `compute` $t_j \leftarrow d^{-1} \cdot v^{-1} \cdot h(m)^{-1}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(t_j) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(T_j = G(t_j) \| d_j \| \mathsf{signT})$
- `Pick` the next β_j in the raPre(n) instance
- `Compute` $a_j \leftarrow \beta_j \cdot v^{-1} \cdot F(R_j)^{-1}$
- `Record`$(R_j \| a_j X \| \mathsf{signR})$
- `Compute` $s_j \leftarrow -F(R_j) \cdot a_j^{-1}$
- `Return`(R_j, T_j, s_j)

Final output:

$a_i - b_i F(R_i) = -a_j b_i \cdot v \cdot F(R_j) \cdot h(m)$
or

$$\begin{aligned} \alpha_i &= \frac{a_i}{b_i} - F(R_i) \\ &= -a_j \cdot v \cdot F(R_j) \cdot h(m^\star) \\ &= -\beta_j h(m^\star) \end{aligned}$$

Advantage and time of $\mathcal{B}$:

$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq \epsilon_1/9 - n^2/q - (m+n)\delta_G - 2mn/q$. and
$t_1 \leq 5n \ln n + m(\tau_{H^1} + \tau_h + 5 \ln n(\tau_{H^2} + \tau_G) + \tau_F)$.

(c) **Type 1:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{group}, \mathsf{signR})$

Procedure:

① In group queries (a_i, b_i):
- plug $\frac{\beta_j}{\alpha_i}$ in $-b_i \cdot v \cdot F(R_i)$.

② In signature queries (m_j, p_j):
- create R_j such that $b_j = 0$.

Group queries (a_i, b_i)**:**

- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` the next (α_i, β_j) in the raPre(n) instance
  `compute` $r_i \leftarrow -{b_i}^{-1} \cdot v^{-1} \cdot \frac{\beta_j}{\alpha_i}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(r_i) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(R_i = G(r_i) \| a_i X + b_i \| \mathsf{group})$
- `Return`(R_i)

Signature queries (m_j, p_j)**:**

- `Compute` $e_p \leftarrow H_v^1(p_j)$
- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` $R_j \xleftarrow{R} S$
  `compute` $d_j \leftarrow H_{e_p}^2(m_j, R_j)$
  `compute` $t_j \leftarrow d^{-1} \cdot v^{-1} \cdot h(m)^{-1}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(t_j) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(T_j = G(t_j) \| d_j \| \mathsf{signT})$
- `Pick` $a_j \xleftarrow{R} S$
- `Record`$(R_j \| a_j X \| \mathsf{signR})$
- `Compute` $s_j \leftarrow -F(R_j) \cdot a_j^{-1}$
- `Return`(R_j, T_j, s_j)

Final output:

$a_j = a_j b_i \cdot v \cdot F(R_i) \cdot h(m^\star)$
or

$$\begin{aligned} 1 &= b_i \cdot v \cdot F(R_i) \cdot h(m^\star) \\ &= -\frac{\beta_j}{\alpha_i} \cdot h(m^\star) \end{aligned}$$

Advantage and time of $\mathcal{B}$:

$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq \epsilon_3/9 - n^2/q - (m+n)\delta_G - 2mn/q$
and
$t_3 \leq \tau + 5n\tau_G \ln n + m(\tau_{H^1} + 5 \ln n(\tau_G + \tau_{H^2}) + \tau_h + \tau_F)$.

(d) **Type 3:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{signR}, \mathsf{group})$

$$\begin{aligned} 1 - \frac{b_i}{a_i} F(R_i) &= d_j \cdot v \cdot F(T_j) h(m^\star) = \lambda_j \cdot h(m^\star) \\ &= d_i \cdot v \cdot F(T_i) h(m_i) \\ &= \lambda_i \cdot h(m_i) \end{aligned}$$

The second equation follows from $1 - \frac{b_i}{a_i} F(R_i) = d_i \cdot v \cdot F(T_i) \cdot h(m_i)$ corresponding to the

Procedure:

① In group queries (a_i, b_i):
- plug $-\beta_i$ in $a_i \cdot F(R_i) \cdot v$.

② In signature queries (m_j, p_j):
- plug α_j in $F(T_j)$.

Group queries (a_i, b_i)**:**

- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` the next β_i in the raPre(n) instance
  `compute` $r_i \leftarrow -\beta_i a_i^{-1} v^{-1}$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(r_i) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(R_i = G(r_i) \| a_i X + b_i \| \mathsf{group})$
- `Return`(R_i)

Signature queries (m_j, p_j)**:**

- `Compute` $e_p \leftarrow H_v^1(p_j)$
- $\mathrm{ctr} \leftarrow 0$
- `Pick` $R_j \xleftarrow{R} S$
- `compute` $d_j \leftarrow H_{e_p}^2(m_j, R_j)$
- `Repeat`
  `pick` the next α_j in the RaPre(n) instance
  `compute` $T_j \leftarrow G(\alpha_j)$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(\alpha_j) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(T_j = G(\alpha_j) \| d_j \| \mathsf{signT})$
- `Pick` $a_j \xleftarrow{R} \mathbb{Z}_q$
- `Compute` $b_j \leftarrow a_j F(R_j)^{-1}(1 + d_j \cdot v \cdot F(T_j) \cdot h(m_j))$
- `Record`$(R_j \| a_j X + b_j \| \mathsf{signR})$
- `Compute` $s_j \leftarrow -F(R_j) \cdot a_j^{-1}$
- `Return`(R_j, T_j, s_j)

Final output:

$-d_j F(T_j) = -d_j a_i \cdot v \cdot F(R_i) \cdot h(m^\star)$
or
$$\begin{aligned} F(T_j) &= a_i \cdot v \cdot F(R_i) \cdot h(m^\star) \\ \alpha_j &= -\beta_i \cdot h(m^\star) \end{aligned}$$

Advantage and time of $\mathcal{B}$**:**

$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq \epsilon_6/9 - n^2/q - (m+n)\delta_G - 2mn/q$
and
$t_6 \leq \tau + 5\tau_G \ln n + m(\tau_{H^1} + \tau_{H^2} + \tau_h + \tau_F + 5\tau_G \ln n)$

(e) **Type 6:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{signT}, \mathsf{group})$

Procedure:

① In signature queries (m_j, p_j):
- plug α_j in $F(T_j)$,
- plug $-\beta_j$ in $a_j F(R_j) v$.

Group queries (a_i, b_i)**:** Do as in 8.4.1.

Signature queries (m_j, p_j)**:**

- `Compute` $e_p \leftarrow H_v^1(p_j)$
- `Pick` $R_j \xleftarrow{R} S$
- `Compute` $d_j \leftarrow H_{e_p}^2(m_j, R_j)$
- $\mathrm{ctr} \leftarrow 0$
- `Repeat`
  `pick` the next α_j in the RaPre(n) instance
  `compute` $T_j \leftarrow G(\alpha_j)$
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$

`Until` $(G(\alpha_j) \neq \mathsf{Fail}) \cup (\mathrm{ctr} = 5 \ln n)$
- `Record`$(T_j = G(\alpha_j) \| d_j \| \mathsf{signT})$
- `Pick` the next β_j in the RaPre(n) instance.
- `Compute` $a_j \leftarrow -\beta_j \cdot v \cdot F(R)^{-1}$
- `Compute` $b_j \leftarrow a_j F(R_j)^{-1}(1 + d_j \cdot v \cdot F(T_j) \cdot h(m_j))$
- `Record`$(R_j \| a_j X + b_j \| \mathsf{signR})$
- `Compute` $s_j \leftarrow -F(R_j) \cdot a_j^{-1}$
- `Return`(R_j, T_j, s_j)

Final output:

$-d_j F(T_j) = -d_j a_i \cdot v \cdot F(R_i) \cdot h(m^\star)$
or
$$\begin{aligned} F(T_j) &= a_i \cdot v \cdot F(R_i) \cdot h(m^\star) \\ \alpha_j &= -\beta_i \cdot h(m^\star) \end{aligned}$$

Advantage and time of $\mathcal{B}$**:**

$\mathbf{Succ}_{h,\mathcal{B}}^{raPre(n)} \geq \epsilon_7/9 - n^2/q - m\delta_G - 2mn/q$
and
$t_7 \leq \tau + n + m(\tau_{H^1} + \tau_{H^2} + \tau_h + 5\tau_G \ln n + \tau_F)$

(f) **Type 7:** $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{signT}, \mathsf{signR})$

equality fulfilled by the signature (R_i, T_i, s_i) on the query (m_i, p_i). It is worth noting that $m_i \neq m^\star$ since the attacker $\mathcal{A}$ is not allowed to return a forgery on a message he has previously queried.

More precisely, on the signature query (m_j, p_j), $\mathcal{C}$ does the following:

- `Compute` $e_p = H_v^1(p_j)$;
- `Pick` $R_j \xleftarrow{R} S$;
- `Compute` $d_j = H_{e_p}^2(m_j, R_j)$;

- $\mathrm{ctr} \leftarrow 0$;
- `Repeat`
  `pick` the next λ_j in the instance of the random linear collision problem rlColl(n);
  `compute` $t_j \leftarrow \lambda_j \cdot d_j^{-1} \cdot v^{-1}$;
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$;
  `Until` $(G(t_j) \neq \mathsf{Fail}) \cup (ctr = 5 \ln n)$;
- `Compute` $\alpha_j = (1 - \lambda_j h(m_j))F(R_j)^{-1}$;
- `Pick` $a_j \stackrel{R}{\leftarrow} \mathbb{Z}_q^{\times}$;
- `Compute` $b_j = a_j \cdot \alpha_j$;
- `Compute` $s_j = (d_j \cdot v \cdot h(m_j) \cdot F(T_j) - 1) \cdot b_j^{-1}$;
- `Record` $(R_j || a_j X + b_j || \mathsf{signR})$;
- `Record` $(T_j || d_j || \mathsf{signT})$;
- `Return` (R_j, T_j, s_j);

It is easy to conclude that this simulation, together with the above forgery returned by the attacker will lead $\mathcal{C}$ to a random linear collision in time t_5:

$$t_5 \leq t + n + m(\tau_{H^1} + \tau_{H^2} + \tau_h + \tau_F + 5\tau_G \ln n)$$

with success probability

$$\mathbf{Succ}_{h,\mathcal{C}}^{rlColl(n)} \geq \epsilon_5/9 - n^2/q - m\delta_G - 2mn/q$$

Description of $\mathcal{D}$. $\mathcal{D}$ exploits a forgery $(R^\star, T^\star, s^\star)$ where $\mathsf{Tag}(R^\star, T^\star) = (\mathsf{group}, \mathsf{group})$ (*i.e.* a **Type 0** forgery) to find a preimage of a certain value, say a. The equation satisfied by the forgery is:

$$a_i - b_i F(R_i) = (a_i b_j - a_j b_i)F(R_j) \cdot v \cdot h(m).$$

To simulate the group oracle, $\mathcal{D}$ selects in advance $i, j \in_R [\![1, n]\!]$. If $i < j$, then on the i-th query (a_i, b_i), $\mathcal{D}$ will select $R_i \in_R S$ and record it using `Record`$(R_i \| a_i X + b_i \| \mathsf{group})$. On the $j-$th query (a_j, b_j), `compute` $R_j \leftarrow G(a \cdot (a_i - b_i F(R_i))(a_i b_j - a_j b_i)^{-1} v^{-1})$. With probability at least $1/n^2$, $\mathcal{D}$ would have chosen the correct i, j and the success of having $R_j \neq \perp$ is at least $1/3$ (almost invertibility of F and randomness of a). If [6] $j < i$, $\mathcal{D}$ will proceed in a similar manner. The remaining queries (a_ℓ, b_ℓ), $\ell \neq i, j$, will be answered exactly as in 8.4.1.
To answer the signature queries (m_j, p_j), $\mathcal{D}$ does the following:

[6] In case $i = j$, we will have $a_i - b_i F(R_i) = 0$, from which $\mathcal{D}$ won't learn anything. In order to prevent such a case, $\mathcal{D}$ must insure that $F(R_i) \neq \frac{a_i}{b_i}$ for the i-th query (a_i, b_i), which is satisfied with probability at least $1 - 1/q$.

- Compute $e_p \leftarrow H_v^1(p_j)$;
- Pick $R_j, T_j \stackrel{R}{\leftarrow} S$;
- Compute $d_j \leftarrow H_{e_p}^2(m_j, R_j)$;
- Compute $\alpha_j = F(R_j)^{-1}(1 - d_j \cdot v \cdot F(T_j) \cdot h(m_j))$;
- Pick $a_j \stackrel{R}{\leftarrow} \mathbb{Z}_q^{\times}$;
- Compute $b_j = a_j \alpha_j$;
- Compute $s_j \leftarrow (d_j \cdot v \cdot t_j \cdot h(m_j) - 1) b_j^{-1}$;
- Record($R_j \| a_j X + b_j \|$signR);
- Record($T_j \| d_j \|$signT);
- Return (R_j, T_j, s_j);
- Compute $e_p \leftarrow H_v^1(p_j)$;
- Pick $R_j \stackrel{R}{\leftarrow} S$;
- Compute $d_j \leftarrow H_{e_p}^2(m_j, R_j)$;
- $\mathrm{ctr} \leftarrow 0$;
- Repeat
  pick $a_j, b_j \stackrel{R}{\leftarrow} \mathbb{Z}_q$;
  compute $t_j \leftarrow (a_j - b_j \cdot F(R_j)) a_j^{-1} d_j^{-1} v^{-1} h(m_j)^{-1}$;
  $\mathrm{ctr} \leftarrow \mathrm{ctr} + 1$;
  Until $(G(t_j) \neq \perp) \cup (\mathrm{ctr} = 5 \ln n)$;
- Compute $s_j \leftarrow (d_j \cdot v \cdot t_j \cdot h(m_j) - 1) b_j^{-1}$;
- Record($R_j \| a_j X + b_j \|$signR);
- Record($T_j = G(t_j) \| d_j \|$signT);
- Return (R_j, T_j, s_j);

We have:

$$\mathbf{Succ}_{h,\mathcal{D}}^{Pre(n)} \geq \frac{\epsilon_0}{54n^2} - 2n^2/q - m\delta_G - 2mn/q.$$

and

$$t_0 \leq t + n + m(\tau_{H^1} + \tau_{H^2} + \tau_h + \tau_F + 5\tau_G \ln n).$$

□

8.4.3 Invisibility

Theorem 8.4. *Let $\mathcal{A}$ be an INV-CMA-adversary operating in time t, after n group queries and m signing queries, with advantage $\epsilon = \mathbf{Succ}^{\textit{inv-cma}}_{\mathsf{US},\mathcal{A}}$, such that $n \gg 2$.*

There exists an adversary $\mathcal{B}$, operating in time t' and attempting to break the pseudo-randomness property of H^1, after m queries (to H^1), with success probability $\mathbf{Succ}^{prf}_{H^1,\mathcal{B}}$ such that:

$$t' \leq t + n + m(\tau_{H^2} + \tau_h + 2\tau_F)$$

and

$$\mathbf{Succ}^{prf}_{H^1,\mathcal{B}_1} \geq \frac{\mathbf{Succ}^{\textit{inv-cma}}_{\mathsf{US},\mathcal{A}}}{2} - \frac{n^2}{2q} - mn/q$$

where τ_F, τ_{H^2} and τ_h are the running time for F, H^2 and h respectively.

Proof. Let $\mathcal{B}$ be the adversary attempting to break the pseudo-randomness of H^1 using an INV-CMA-adversary $\mathcal{A}$ against the above undeniable signatures. $\mathcal{A}$ operates as previously in the generic group model, and the polynomials F_ℓ manipulated by $\mathcal{B}$ are also affine univariate, i.e. of the form $aX+b$, however the indeterminate refers to the public key V. In fact, $\mathcal{B}$ does not know v (otherwise his task would be easy), but is allowed to choose the private key u.

Let $m_0^\star, m_1^\star$, and $p^\star$ be the challenge messages and the challenge time period resp. $\mathcal{B}$ will forward $p^\star$ to his own challenger as a challenge seed and will receive a string $e^\star$ which is either the result of applying H^1_v to $p^\star$ or a uniformly chosen random string from the corresponding space. $\mathcal{B}$ will then form the challenge signature $\mu^\star = (R^\star, T^\star, s^\star)$, using $e^\star$, on the message $m_b^\star$ for $b \xleftarrow{R} \{0,1\}$. If $e^\star = H^1_v(p^\star)$, then $\mu^\star$ is valid signature on $m_b^\star$, otherwise it is an invalid signature on both $m_0^\star$ and $m_1^\star$. Thus, the answer of $\mathcal{A}$ will suffice $\mathcal{B}$ to conclude.

More precisely, $\mathcal{B}$ will proceed as follows:

Game 0. Let $m_0^\star, m_1^\star$ and $p^\star$ be the challenge messages and the challenge time period resp. $\mathcal{B}$ will form an undeniable signature $\mu^\star$, following the standard signing algorithm, on $m_b^\star$ for some $b \xleftarrow{R} \{0,1\}$. We denote by S_0 be the event "$\mathcal{A}$ returns the bit b" and we use a similar notation S_i in any **Game** i. By definition, we have $\Pr[S_0] = \epsilon + \frac{1}{2}$.

Game 1. $\mathcal{B}$ uses the interpretation described above which considers a safe sequence in order to simulate the group oracle. We get:

$$|\Pr[S_1] - \Pr[S_0]| \leq (n+4)^2/q$$

Game 2. In this game, $\mathcal{B}$ simulates the signing oracle. Let (m, p) be the signing query where m denotes the message to be signed and $p \neq p^\star$ denotes the time period. A signature (R, T, s) on m for the time period p should satisfy:

$$(d \cdot F(T) \cdot h(m))V = F(R)U + sR + P$$

where $T = dP$ and $d = H^2_{e_p}(m, R)$ and $e_p = H^1_v(p)$.

Thus if we write $R = aV + bP$, we get:

$$\begin{aligned} d \cdot F(T) \cdot h(m) &= s \cdot a, \\ 0 &= uF(R) + s \cdot b + 1. \end{aligned}$$

Thus, (R, T, s) should satisfy:

$$ab^{-1}(uF(R) + 1) = -d \cdot F(T) \cdot h(m).$$

As a consequence, $\mathcal{B}$ will do the following:

- request his challenger for $e_p = H^1_v(p)$,
- pick $R, T \xleftarrow{R} S$ and compute $d = H^2_{e_p}(m, R)$,
- pick $b \xleftarrow{R} \mathbb{Z}_q^\times$ and compute $a = -b \cdot d \cdot F(T) \cdot h(m) \cdot (uF(R) + 1)^{-1}$,
- execute $\mathtt{Record}(T \| d \| \mathsf{signT})$ and $\mathtt{Record}(R \| aX + b \| \mathsf{signR})$.

The difference between the previous game is when the introduction of R and T along with their polynomials leads to inconsistencies in simulating the group oracle, i.e. collisions with polynomials in $\mathcal{F}$. The probability that these collisions occur is upper-bounded by $2n/q$, thus:

$$|\Pr[S_2] - \Pr[S_1]| \leq 2mn/q$$

Game 3. In this game, $\mathcal{B}$ simulates the challenge signature generation; he proceeds exactly as in **Game 2**. The difference is when the created $R^\star$ and $T^\star$ (elements of the challenge signature) lead to inconsistencies with the group oracle:

$$|\Pr[S_3] - \Pr[S_2]| \leq 2n/q$$

Game 4. In this game, $\mathcal{B}$ simulates the verification and conversion oracles. Since verification and conversion queries can occur only with respect to time periods $p \neq p^\star$, $\mathcal{B}$ can request his challenger for the conversion receipt $e_p = H^1_v(p)$ for the time period p, and simulate perfectly the verification/conversion oracles. We clearly have $\Pr[S_4] = \Pr[S_3]$.

Game 5. In this game, we modify the challenge signature generation. In fact, after $\mathcal{A}$ outputs $m_0^\star, m_1^\star$, and $p^\star$, $\mathcal{B}$ outputs $p^\star$ to his own challenger as a challenge seed, and gets a challenge bit-string $e^\star$, which is either $H^1_v(p^\star)$, if some $b' \xleftarrow{R} \{0, 1\}$ is 1, or a random string from the given space otherwise. $\mathcal{B}$ produces then the challenge signature $\mu^\star = (R^\star, T^\star, s^\star)$ on $m_b^\star$

using $e^\star$, i.e. proceeds exactly as the standard algorithm with the exception of computing $d^\star$ as $H^2_{e^\star}(m^\star_b, R^\star)$. Note that when $e^\star$ is a random string, then $\mu^\star$ is not a valid signature on neither $m^\star_0$ nor $m^\star_1$. Clearly:

$$\Pr[S_5] = \Pr[b_a = b | b' = 1]$$

and

$$\Pr[b_a \neq b | b' = 0] = \frac{1}{2}$$

At the end of the simulation, if $\mathcal{A}$ outputs $b_a = b$, then $\mathcal{B}$ will respond $b" = 1$, i.e. $e^\star$ is indeed $H^1_v(p^\star)$, otherwise he responds $b" = 0$. We have:

$$\begin{aligned}
\mathbf{Succ}^{prf}_{H^1,\mathcal{B}} &= \left|\Pr[b" = b'] - \frac{1}{2}\right| \\
&= \left|\Pr[b" = 1, b' = 1] + \Pr[b" = 0, b' = 0] - \frac{1}{2}\right| \\
&= \left|\Pr[b" = 1 | b' = 1]\Pr[b' = 1] + \Pr[b" = 0 | b' = 0]\Pr[b' = 0] - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|\Pr[b" = 1 | b' = 1] + \Pr[b" = 0 | b' = 0] - 1\right| \\
&= \frac{1}{2}\left|\Pr[b_a = b | b' = 1] + \Pr[b_a \neq b | b' = 0] - 1\right| \\
&= \frac{1}{2}\left|\Pr[S_5] - \frac{1}{2}\right| \\
&\geq \frac{\epsilon}{2} - \frac{n^2}{2q} - mn/q
\end{aligned}$$

Moreover,

$$t' \leq t + n + m(\tau_{H^2} + \tau_h + 2\tau_F)$$

□

8.5 Conclusion

We properly defined security notions for convertible undeniable signatures that support the additional property of *achronous* gradual conversion. Adapting the scheme proposed by Michels, Petersen, and Horster in 1996, we realized the first scheme featuring this useful notion of conversion.

In addition, we gave the first security analysis of the Michels-Petersen-Horster protocol, thereby addressing a problem left open since 1996. We have modified this scheme such that it becomes a generic one, which allows to use it for instance in the setting of elliptic curves (and therefore offers attractive practical advantages in terms of signature length and performances). In this context and in comparison with the time-selective convertible undeniable signatures from [Laguillaumie & Vergnaud, 2005], the computational costs for the confirmation/disavowal protocols and the conversion algorithms are much smaller.

Conclusion

In this thesis, we were interested in signatures with controlled verification, more specifically undeniable and confirmer signatures. We actually focused on how to produce these signatures from basic cryptographic primitives such as digital signatures, encryption, and commitment schemes. In fact, we noticed that even the monolithic realizations of these signatures are built upon popular primitives, which results in security and efficiency analyses similar to those of the underlying components, but still indispensable to carry out. Our main purpose was to understand then bridge the gap between these realizations and the known generic constructions of such opaque signatures.

To analyze the generic constructions of confirmer signatures, we used the famous *meta-reduction* tool; such a tool was mainly applied to achieve impossibility results, e.g. disproving equivalence between complexity assumptions or separating results between idealized and standard models. In our study, we used meta-reductions to show that the popular generic constructions cannot achieve secure confirmer signatures without using strong encryption as a building block, which engenders expensive confirmer signatures with limited efficient instantiations. This is actually due to an inherent weakness in these constructions that consists in the possibility of creating confirmer signatures without the help of the signer. After identifying the weaknesses in the popular generic constructions, comes the task of annihilating these weaknesses at cheap costs and without compromising the security. Fortunately, this was doable by simply binding the digital signature - these generic constructions require always the computation of a digital signature - to the resulting confirmer signature. The outcome of this tweak was tremendous as it made the constructions rest on very cheap encryption, and consequently led to short confirmer signatures with small generation, verification, and conversion costs. Another important consequence of this slight change consists in allowing *homomorphic encryption* in the design, which translates in efficient confirmation and denial protocols.

The immediate prospect of such an analysis is its extension to other opaque or privacy-preserving mechanisms/signatures, e.g. group signatures, designated verifier signatures, or anonymous credentials. In fact, most such mechanisms involve a digital signature on some message and an encryption layer that ensures the privacy. Hence the possibility of applying the same techniques in order to allow cheap and useful encryption in the design, and thus achieve constructions with many efficient instantiations. The long-run prospect consists in systematically applying the meta-reduction tool in other cryptographic realizations in order to spot the potential flaws in the design, and later repair these flaws and improve the resulting constructions.

Bibliography

AGGARWAL, D. & MAURER, U. M. (2009). Breaking RSA Generically Is Equivalent to Factoring. In: Joux [2009a], pp. 36–53.

AN, J. H., DODIS, Y. & RABIN, T. (2002). On the Security of Joint Signature and Encryption. In: Knudsen [2002], pp. 83–107.

BAO, F., DENG, R. H. & ZHOU, J. (eds.) (2004). *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, vol. 2947 of *LNCS*. Springer.

BARIĆ, N. & PFITZMANN, B. (1997). Collision-Free Accumulators and Fail-Stop Signature Schemes. In: Fumy [1997], pp. 480–494.

BELLARE, M., BOLDYREVA, A., DESAI, A. & POINTCHEVAL, D. (2001). Key-Privacy in Public-Key Encryption. In: *Advances in Cryptology - ASIACRYPT 2001* (BOYD, C., ed.), vol. 2248 of *LNCS*. Springer.

BELLARE, M., DESAI, A., POINTCHEVAL, D. & ROGAWAY, P. (1998). Relations Among Notions of Security for Public-Key Encryption Schemes. In: *Advances in Cryptology - CRYPTO'98* (KRAWCZYK, H., ed.), vol. 1462 of *LNCS*. Springer.

BELLARE, M. & KOHNO, T. (2004). Hash Function Balance and Its Impact on Birthday Attacks. In: Cachin & Camenisch [2004], pp. 401–418.

BELLARE, M. & ROGAWAY, P. (1993). Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: *Proceedings of the First ACM Conference on Computer and Communications Security* (DENNING, D., PYLE, R., GANESAN, R., SANDHU, R. & ASHBY, V., eds.). ACM Press.

BELLARE, M. & ROGAWAY, P. (1996). The Exact Security of Digital Signatures: How to Sign with RSA and Rabin. In: Maurer [1996], pp. 399–416.

BIEHL, I., PAULUS, S. & TAKAGI, T. (2004). Efficient Undeniable Signature Schemes based on Ideal Arithmetic in Quadratic Orders. *Des. Codes Cryptography* **31**(2), 99–123.

BLAKE, I. F., SEROUSSI, G. & SMART, N. (2005). *Advances in Elliptic Curve Cryptography*. Cambrige University Press.

BLANCHET, B. & POINTCHEVAL, D. (2006). Automated Security Proofs with Sequences of Games. In: *CRYPTO* (DWORK, C., ed.), vol. 4117 of *LNCS*. Springer.

BLUM, L., BLUM, M. & SHUB, M. (1986). A Simple Unpredictable Pseudo-Random Number Generator. *SIAM J. Comput.* **15**(2), 364–383.

BLUM, M., FELDMAN, P. & MICALI, S. (1988). Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In: *Proceedings of the 20th ACM Symposium on Theory of Computing (STOC'88)* (SIMON, J., ed.). ACM Press.

BONEH, D. & BOYEN, X. (2004). Short Signatures Without Random Oracles. In: Cachin & Camenisch [2004], pp. 56–73.

BONEH, D., BOYEN, X. & SHACHAM, H. (2004a). Short Group Signatures. In: Franklin [2004], pp. 41–55.

BONEH, D., LYNN, B. & SHACHAM, H. (2004b). Short Signatures from the Weil Pairing. *J. Cryptology* **17**(4), 297–319.

BONEH, D. & VENKATESAN, R. (1998). Breaking RSA May Not Be Equivalent to Factoring. In: Nyberg [1998], pp. 59–71.

BOYAR, J., CHAUM, D., DAMGÅRD, I. B. & PEDERSEN, T. B. (1991). Convertible undeniable signatures. In: *Advances in Cryptology* - CRYPTO*'90* (MENEZES, A. J. & VANSTONE, S. A., eds.), vol. 537 of *LNCS*. Springer.

BOYD, C. & FOO, E. (1998). Off-line Fair Payment Protocols using Convertible Signatures. In: *Advances in Cryptology* - ASIACRYPT*'98* (OHTA, K. & PEI, D., eds.), vol. 1514 of *LNCS*. Springer.

BRASSARD, G. (ed.) (1990). *Advances in Cryptology* - CRYPTO *'89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, vol. 435 of *LNCS*. Springer.

BRASSARD, G., CHAUM, D. & CRÉPEAU, C. (1988). Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* **37**(2), 156–189.

BROWN, D. R. L. (2005). Generic Groups, Collision Resistance, and ECDSA. *Des. Codes Cryptography* **35**(1), 119–152.

CACHIN, C. & CAMENISCH, J. (eds.) (2004). *Advances in Cryptology* - EUROCRYPT *2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, vol. 3027 of *LNCS*. Springer.

CAMENISCH, J., CHANDRAN, N. & SHOUP, V. (2009). A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux [2009a], pp. 351–368.

CAMENISCH, J. & LYSYANSKAYA, A. (2002). Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung [2002], pp. 61–76.

CAMENISCH, J. & LYSYANSKAYA, A. (2004). Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin [2004], pp. 56–72.

CAMENISCH, J. & MICHELS, M. (2000). Confirmer Signature Schemes Secure against Adaptative Adversaries. In: Preneel [2000], pp. 243–258.

CAMENISCH, J. & SHOUP, V. (2003). Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: *Advances in Cryptology* - CRYPTO *2003* (BONEH, D., ed.), vol. 2729 of *LNCS*. Springer.

CASTAGNOS, G., JOUX, A., LAGUILLAUMIE, F. & NGUYEN, P. Q. (2009). Factoring q^2 with Quadratic Forms: Nice Cryptanalyses. In: ASIACRYPT *2009* (MATSUI, M., ed.), vol. 5912 of *LNCS*. Springer.

CASTAGNOS, G. & LAGUILLAUMIE, F. (2009). On the Security of Cryptosystems with Quadratic Decryption: The Nicest Cryptanalysis. In: Joux [2009a], pp. 260–277.

CHAUM, D. (1991a). Some Weaknesses of "Weaknesses of Undeniable Signatures". In: Davies [1991], pp. 205–220.

CHAUM, D. (1991b). Zero-Knowledge Undeniable Signatures. In: *Advances in Cryptology* - EUROCRYPT*'90* (DAMGÅRD, I., ed.), vol. 473 of *LNCS*. Springer.

CHAUM, D. (1995). Designated Confirmer Signatures. In: *Advances in Cryptology* - EUROCRYPT*'94* (DE SANTIS, A., ed.), vol. 950 of *LNCS*. Springer.

CHAUM, D. & PEDERSEN, T. P. (1993). Wallet Databases with Observers. In: *Advances in Cryptology* - CRYPTO*'92* (BRICKELL, E. F., ed.), vol. 740 of *LNCS*. Springer.

CHAUM, D. & VAN ANTWERPEN, H. (1990). Undeniable Signatures. In: Brassard [1990], pp. 212–216.

CHAUM, D., VAN HEIJST, E. & PFITZMANN, B. (1991). Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer. In: *CRYPTO* (FEIGENBAUM, J., ed.), vol. 576 of *LNCS*. Springer.

CHEN, T.-S., HUANG, J.-Y. & CHEN, T.-L. (2005). An efficient undeniable group-oriented signature scheme. *Applied Mathematics and Computation* **165**(1), 95–102.

COHEN, H. (1996). *A course in computational algebraic number theory.* Graduate Texts in Mathematics 138. Springer-Verlag.

COHEN, H. & FREY, G. (eds.) (2005). *Handbokk of Elliptic and Hyperelliptic Curve Cryptography*, vol. 34 of Discrete Mathematics and its Applications. Chapman & Hall/CRC. Taylor and Francis .

CORON, J.-B. & ICART, T. (2009). An indifferentiable hash function into elliptic curves. Cryptology ePrint Archive, Report 2009/340. `http://eprint.iacr.org/`.

CORON, J.-S. (2002). Optimal Security Proofs for PSS and Other Signature Schemes. In: Knudsen [2002], pp. 272–287.

CORON, J.-S., DODIS, Y., MALINAUD, C. & PUNIYA, P. (2005). Merkle-Damgård Revisited: How to Construct a Hash Function. In: *CRYPTO* (SHOUP, V., ed.), vol. 3621 of *LNCS*. Springer.

CORON, J.-S., PATARIN, J. & SEURIN, Y. (2008). The random oracle model and the ideal cipher model are equivalent. In: *CRYPTO* (WAGNER, D., ed.), vol. 5157 of *LNCS*. Springer.

CRAMER, R. (ed.) (2005). *Advances in Cryptology* - EUROCRYPT *2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, vol. 3494 of *LNCS*. Springer.

CRAMER, R. & SHOUP, V. (2000). Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.* **3**(3), 161–185.

CRAMER, R. & SHOUP, V. (2003). Design and Analysis of Practical Public-Key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.* **33**(1), 167–226.

DAMGÅRD, I. B. & PEDERSEN, T. P. (1996). New Convertible Undeniable Signature Schemes. In: Maurer [1996], pp. 372–386.

DAMGÅRD, I. (1989). A Design Principle for Hash Functions. In: Brassard [1990], pp. 416–427.

DAMGÅRD, I. (2000). Efficient concurrent zero-knowledge in the auxiliary string model. In: Preneel [2000], pp. 418–430.

DAMGÅRD, I., FAZIO, N. & NICOLOSI, A. (2006). Non-interactive zero-knowledge from homomorphic encryption. In: TCC *2006* (HALEVI, S. & RABIN, T., eds.), vol. 3876 of *LNCS*. Springer.

DAVIES, D. W. (ed.) (1991). *Advances in Cryptology* - EUROCRYPT*'91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, vol. 547 of *LNCS*. Springer.

DENT, A. W. (2002). Adapting the Weaknesses of the Random Oracle Model to the Generic Group Model. In: *Advances in Cryptology* - ASIACRYPT *2002* (ZHENG, Y., ed.), vol. 2501 of *LNCS*. Springer.

DESMEDT, Y. & YUNG, M. (1991). Weakness of Undeniable Signature Schemes (Extended Abstract). In: Davies [1991], pp. 205–220.

DIFFIE, W. & HELLMAN, M. E. (1976). New Directions in Cryptography. *IEEE Trans. Inf. Theory* **22**, 644–654.

DOBBERTIN, H. (1996). Cryptanalysis of MD4. In: *FSE1996* (GOLLMANN, D., ed.), vol. 1039 of *LNCS*. Springer.

DOLEV, D., DWORK, C. & NAOR, M. (1991). Non-Malleable Cryptography (Extended Abstract). In: *STOC*. ACM.

EL AIMANI, L. (2008). Toward a Generic Construction of Universally Convertible Undeniable Signatures from Pairing-Based Signatures. In: *Progress in Cryptology* - INDOCRYPT *2008* (CHOWDHURY, D. R., RIJMEN, V. & DAS, A., eds.), vol. 5365 of *LNCS*. Springer. Full version available at the Cryptology ePrint Archive, Report 2009/362.

EL AIMANI, L. (2009a). Anonymity from Public Key Encryption to Undeniable Signatures. In: Preneel [2009], pp. 217–234.

EL AIMANI, L. (2009b). On Generic Constructions of Designated Confirmer Signatures. In: *Progress in Cryptology* - INDOCRYPT *2009* (ROY, B. & SENDRIER, N., eds.), vol. 5922. Berlin, Heidelberg. Full version available at the Cryptology ePrint Archive, Report 2009/403.

EL AIMANI, L. (2010). Efficient Confirmer Signature from the "Signature of a Commitment" Paradigm. In: PROVSEC *2010* (HENG, S.-H. & KUROSAWA, K., eds.), vol. 6402 of *LNCS*. Springer. Full version available at the Cryptology ePrint Archive, Report 2009/435.

EL AIMANI, L. & RAEKOW, R. (2009). Exploring Subliminal Channels in Pairing-Based Signatures. WEWoRC 2009.

EL AIMANI, L. & RAEKOW, Y. (2010). Reselling Digital Content. In: FARES *2010* (O'CONNER, L., ed.). 10662 Los Vaqueros Circle Los Alamitos, California 90720-1314: IEEE Computer Society.

EL AIMANI, L. & VERGNAUD, D. (2007). Gradually Convertible Undeniable Signatures. In: *ACNS'07* (KATZ, J. & YUNG, M., eds.), vol. 4521 of *LNCS*. Springer.

EL AIMANI, L. & VON ZUR GATHEN, J. (2007). Finding Low Weight Polynomial Multiples Using Lattices . Poster session of the LLL + 25 conference. Full version available at the Cryptology ePrint Archive, Report 2007/423.

EL GAMAL, T. (1985). A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Trans. Inf. Theory* **31**, 469–472.

FIAT, A. & SHAMIR, A. (1986). How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: *CRYPTO* (ODLYZKO, A. M., ed.), vol. 263 of *LNCS*. Springer.

FRANKLIN, M. K. (ed.) (2004). *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, vol. 3152 of *LNCS*. Springer.

FUJIOKA, A., OKAMOTO, T. & OHTA, K. (1991). Interactive Bi-Proof Systems and Undeniable Signature Schemes. In: Davies [1991], pp. 243–256.

FUMY, W. (ed.) (1997). *Advances in Cryptology* - EUROCRYPT*'97, International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, Germany, May 11-15, 1997, Proceeding*, vol. 1233 of *LNCS*. Springer.

GALBRAITH, S. D. & MAO, W. (2003). Invisibility and Anonymity of Undeniable and Confirmer Signatures. In: *Topics in Cryptology* - CT-RSA *2003* (JOYE, M., ed.), vol. 2612 of *LNCS*. Springer.

GALBRAITH, S. D., MAO, W. & PATERSON, K. G. (2002). RSA-based Undeniable Signatures for General Moduli. In: *Topics in Cryptology* - CT-RSA *2002* (PRENEEL, B., ed.), vol. 2271 of *LNCS*. Springer.

GALINDO, D., HERRANZ, J. & KILTZ, E. (2006). On the Generic Construction of Identity-Based Signatures with Additional Properties. In: Lai & Chen [2006], pp. 178–193.

GENNARO, R., HALEVI, S. & RABIN, T. (1999). Secure Hash-and-Sign Signatures Without the Random Oracle. In: Stern [1999], pp. 397–416.

GENNARO, R., RABIN, T. & KRAWCZYK, H. (2000). RSA-Based Undeniable Signatures. *J. Cryptology* **13**(4), 397–416.

GENTRY, C., MOLNAR, D. & RAMZAN, Z. (2005). Efficient Designated Confirmer Signatures Without Random Oracles or General Zero-Knowledge Proofs. In: Roy [2005], pp. 662–681.

GOLDREICH, O. (2001). *Foundations of cryptography. Basic Tools.* Cambridge University Press.

GOLDREICH, O., MICALI, S. & WIGDERSON, A. (1991). Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *J. Assoc. Comput. Mach.* **38**(3), 691–729.

GOLDREICH, O., SAHAI, A. & VADHAN, S. P. (1998). Honest-Verifier Statistical Zero-Knowledge Equals General Statistical Zero-Knowledge. In: *STOC*.

GOLDWASSER, S. (2009). Cryptography without (Hardly Any) Secrets ? In: Joux [2009a], pp. 369–370.

GOLDWASSER, S. & MICALI, S. (1984). Probabilistic Encryption. *J. Comput. Syst. Sci.* **28**, 270–299.

GOLDWASSER, S., MICALI, S. & RACKOFF, C. (1989). The Knowledge Complexity of Interactive Proof-Systems. *SIAM J. Comput.* **18**(1), 186–206.

GOLDWASSER, S., MICALI, S. & RIVEST, R. L. (1988). A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* **17**(2), 281–308.

GOLDWASSER, S. & TAUMAN KALAI, Y. (2003). On the (In)security of the Fiat-Shamir Paradigm. In: *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS 2003)* (SUDAN, M., ed.). IEEE Computer Society.

GOLDWASSER, S. & WAISBARD, E. (2004). Transformation of Digital Signature Schemes into Designated Confirmer Signature Schemes. In: *Theory of Cryptography, TCC 2004* (NAOR, M., ed.), vol. 2951 of *LNCS*. Springer.

GROTH, J. & SAHAI, A. (2008). Efficient Non-interactive Proof Systems for Bilinear Groups. In: EUROCRYPT *2008* (SMART, N. P., ed.), vol. 4965 of *LNCS*. Springer.

GUO, X.-Y. & TANG, C.-J. (2005). Identity based group-oriented undeniable signature scheme. *Applied Mathematics and Computation* **169**(2), 1448–1457.

HALEVI, S. (2005). A sufficient condition for key-privacy. Available at http://eprint.iacr.org/2005/005.

HAN, S., CHANG, E., DENG, X., YEUNG, W. K. Y. & GAO, L. (2004). Practical Fair Anonymous Undeniable Signatures. In: *International Conference on Computational Intelligence* (OKATAN, A., ed.). International Computational Intelligence Society.

HAN, S., CHANG, E., DILLON, T. S. & WANG, J. (2006). Improvement of a Convertible Undeniable Partially Blind Signature Scheme. In: *AINA (1)*. IEEE Computer Society.

HARN, L. & YANG, S. (1993). Group-oriented Undeniable Signature Schemes without the Assistance of a Mutually Trusted Party. In: Seberry & Zheng [1993], pp. 133–142.

HERRANZ, J., HOFHEINZ, D. & KILTZ, E. (2006). KEM/DEM: Necessary and Sufficient Conditions for secure Hybrid Encryption. Available at http://eprint.iacr.org/2006/265.pdf.

HOFHEINZ, D. & KILTZ, E. (2009). Practical Chosen Ciphertext Secure Encryption from Factoring. In: Joux [2009a], pp. 313–332.

HOHENBERGER, S. & WATERS, B. (2009). Short and Stateless Signatures from the RSA Assumption. In: *CRYPTO* (HALEVI, S., ed.), vol. 5677 of *LNCS*. Springer.

HUANG, Q. & WONG, D. S. (2009). New Constructions of Convertible Undeniable Signature Schemes without Random Oracles. Cryptology ePrint Archive, Report 2009/517. `http://eprint.iacr.org/`.

HUANG, X., MU, Y., SUSILO, W. & WU, W. (2007a). A Generic Construction for Universally-Convertible Undeniable Signatures. In: *CANS* (BAO, F., LING, S., OKAMOTO, T., WANG, H. & XING, C., eds.), vol. 4856 of *LNCS*. Springer.

HUANG, X., MU, Y., SUSILO, W. & WU, W. (2007b). Provably Secure Pairing-Based Convertible Undeniable Signature with Short Signature Length. In: *Pairing* (TAKAGI, T., OKAMOTO, T., OKAMOTO, E. & OKAMOTO, T., eds.), vol. 4575 of *LNCS*. Springer.

HUANG, Z., CHEN, Z. & WANG, Y. (2005). Convertible Undeniable Partially Blind Signatures. In: *AINA*. IEEE Computer Society.

JAKOBSSON, M. (1994). Blackmailing using undeniable signatures. In: *EUROCRYPT*.

JAKOBSSON, M., SAKO, K. & IMPAGLIAZZO, R. (1996). Designated Verifier Proofs and Their Applications. In: Maurer [1996], pp. 143–154.

JOUX, A. (ed.) (2009a). *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, vol. 5479 of *LNCS*. Springer.

JOUX, A. (2009b). *Algorithmic Cryptanalyis*. Chapman & Hall/CRC. Taylor and Francis .

KERCKHOFFS, A. (1883). La Cryptographie Militaire. *Journal des sciences militaires* **IX**, 161–191. Avaiable at http://petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf.

KIM, S. & WON, D. (2004). Threshold Entrusted Undeniable Signature. In: *ICISC* (PARK, C. & CHEE, S., eds.), vol. 3506 of *LNCS*. Springer.

KLEINJUNG, T., AOKI, K., FRANKE, J., LENSTRA, A., THOMÉ, E., BOS, J., GAUDRY, P., KRUPPA, A., MONTGOMERY, P., OSVIK, D. A., TE RIELE, H., ANDREY TIMOFEEV, A. & ZIMMERMANN, P. (2010). Factorization of a 768-bit RSA modulus. Cryptology ePrint Archive, Report 2010/006. `http://eprint.iacr.org/`.

KNUDSEN, L. R. (ed.) (2002). *Advances in Cryptology* - EUROCRYPT *2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, vol. 2332 of *LNCS*. Springer.

KOIDE, A., TSO, R. & OKAMOTO, E. (2008). Convertible Undeniable Partially Blind Signature from Bilinear Pairings. In: *EUC (2)* (XU, C.-Z. & GUO, M., eds.). IEEE Computer Society.

KUDLA, C. & PATERSON, K. G. (2005). Non-interactive Designated Verifier Proofs and Undeniable Signatures. In: *Cryptography and Coding, 10th IMA International Conference* (SMART, N. P., ed.), vol. 3796 of *LNCS*. Springer.

KUROSAWA, K. & FURUKAWA, J. (2008). Universally Composable Undeniable Signature. In: *ICALP (2)* (ACETO, L., DAMGÅRD, I., GOLDBERG, L. A., HALLDÓRSSON, M. M., INGÓLFSDÓTTIR, A. & WALUKIEWICZ, I., eds.), vol. 5126 of *LNCS*. Springer.

KUROSAWA, K. & HENG, S.-H. (2005). 3-Move Undeniable Signature Scheme. In: Cramer [2005], pp. 181–197.

KUROSAWA, K. & HENG, S.-H. (2006). Relations Among Security Notions for Undeniable Signature Schemes. In: *SCN* (PRISCO, R. D. & YUNG, M., eds.), vol. 4116 of *LNCS*. Springer.

KUROSAWA, K. & TAKAGI, T. (2006). New Approach for Selectively Convertible Undeniable Signature Schemes. In: Lai & Chen [2006], pp. 428–443.

LAGUILLAUMIE, F. & VERGNAUD, D. (2005). Time-Selective Convertible Undeniable Signatures. In: *Topics in Cryptology* - CT-RSA *2005* (MENEZES, A. J., ed.), vol. 3376 of *LNCS*. Springer.

LAI, X. & CHEN, K. (eds.) (2006). *Advances in Cryptology* - ASIACRYPT *2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shangai, China, December 3-7, 2006, Proceedings*, vol. 4284 of *LNCS*. Springer.

LE TRIEU, P., KUROSAWA, K. & OGATA, W. (2009). New RSA-Based (Selectively) Convertible Undeniable Signature Schemes. In: Preneel [2009], pp. 116–134.

LE TRIEU, P., KUROSAWA, K. & OGATA, W. (2010). Provably Secure Convertible Undeniable Signatures with Unambiguity. In: SCN *2010* (GARAY, J. A. & PRISCO, R. D., eds.), vol. 6480 of *LNCS*. Springer. Full version available at the Cryptology ePrint Archive, Report 2009/394.

LEE, N.-Y. & HWANG, T. (1999). Group-oriented undeniable signature schemes with a trusted center. *Computer Communications* **22**(8), 730–734.

LI, Z., CHONG, C. F., HUI, L. C. K., YIU, S.-M., CHOW, K. P., TSANG, W. W., CHAN, H. W. & PUN, K. K. H. (2007). An attack on libert et al.' s id-based undeniable signature scheme. *I. J. Network Security* **5**(2), 220–223.

LIBERT, B. & QUISQUATER, J.-J. (2004). Identity Based Undeniable Signatures. In: *Topics in Cryptology* - CT-RSA *2004* (OKAMOTO, T., ed.), vol. 2964 of *LNCS*. Springer.

LIM, C. H. & LEE, P. J. (1993). Modified Maurer-Yacobi's scheme and its applications. In: Seberry & Zheng [1993], pp. 308–323.

LIN, C.-H., WANG, ., C-T & CHANG, C.-C. (1996). A group-oriented (t, n) undeniable signature scheme without trusted center. In: ACISP (PIEPRZYK, J. & SEBERRY, J., eds.), vol. 1172 of *LNCS*. Springer.

LU, R., CAO, Z. & ZHOU, Y. (2005). Threshold undeniable signature scheme based on conic. *Applied Mathematics and Computation* **162**(1), 165–177.

LYUU, Y.-D. & WU, M.-L. (2002). Convertible Group Undeniable Signatures. In: *ICISC* (LEE, P. J. & LIM, C. H., eds.), vol. 2587 of *LNCS*. Springer.

MAO, W. (2008). *Modern Cryptography: Theory & Practice*. Dorling Kindersley (India).

MAURER, U. M. (ed.) (1996). *Advances in Cryptology* - EUROCRYPT*'96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, vol. 1070 of *LNCS*. Springer.

MAURER, U. M. & WOLF, S. (1998). Lower Bounds on Generic Algorithms in Groups. In: Nyberg [1998], pp. 72–84.

MICHELS, M., PETERSEN, H. & HORSTER, P. (1996). Breaking and Repairing a Convertible Undeniable Signature Scheme. In: *Proceedings of the Third ACM Conference on Computer and Communications Security* (GONG, L. & STERN, J., eds.). ACM Press.

MICHELS, M. & STADLER, M. (1998). Generic Constructions for Secure and Efficient Confirmer Signature Schemes. In: Nyberg [1998], pp. 406–421.

MILLER, V. S. (2004). The Weil Pairing, and Its Efficient Calculation. *J. Cryptology* **17**(4), 235–261.

MIYAZAKI, T. (2000). An Improved Scheme of the Gennaro-Krawczyk-Rabin Undeniable Signature System Based on RSA. In: *ICISC* (WON, D., ed.), vol. 2015 of *LNCS*. Springer.

MONNERAT, J., OSWALD, Y. A. & VAUDENAY, S. (2005). Optimization of the MOVA Undeniable Signature Scheme. In: *Progress in Cryptology - Mycrypt 2005* (DAWSON, E. & VAUDENAY, S., eds.), vol. 3715 of *LNCS*. Springer.

MONNERAT, J. & VAUDENAY, S. (2004a). Generic Homomorphic Undeniable Signatures. In: *Advances in Cryptology - ASIACRYPT 2004* (LEE, P. J., ed.), vol. 3329 of *LNCS*. Springer.

MONNERAT, J. & VAUDENAY, S. (2004b). Undeniable Signatures Based on Characters: How to Sign with One Bit. In: Bao *et al.* [2004], pp. 69–85.

MONNERAT, J. & VAUDENAY, S. (2005). Chaum's Designated Confirmer Signature Revisited. In: *Information Security, ISC 2005* (ZHOU, J., LOPEZ, J., DENG, R. H. & BAO, F., eds.), vol. 3650 of *LNCS*. Springer.

NECHAEV, V. I. (1994). Complexity of a Determinate Algorithm for the Discrete Logarithm. *Math. Notes* **55**(2), 165–172.

NGUYEN, K. Q., MU, Y. & VARADHARAJAN, V. (1999). Undeniable Confirmer Signature. In: *ISW* (MAMBO, M. & ZHENG, Y., eds.), vol. 1729 of *LNCS*. Springer.

NYBERG, K. (ed.) (1998). *Advances in Cryptology - EUROCRYPT'98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, vol. 1403 of *LNCS*. Springer.

OGATA, W., KUROSAWA, K. & HENG, S.-H. (2005). The Security of the FDH Variant of Chaum's Undeniable Signature Scheme. In: *8th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2005* (VAUDENAY, S., ed.), vol. 3386 of *LNCS*. Springer.

OKAMOTO, T. (1994). Designated Confirmer Signatures and Public-Key Encryption are Equivalent. In: *Advances in Cryptology - CRYPTO'94* (DESMEDT, Y., ed.), vol. 839 of *LNCS*. Springer.

OKAMOTO, T. & POINTCHEVAL, D. (2001). The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In: *4th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2001* (KIM, K., ed.), vol. 1992 of *LNCS*. Springer.

PAILLIER, P. (1999). Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern [1999], pp. 223–238.

PAILLIER, P. (2007). Impossibility Proofs for RSA Signatures in the Standard Model. In: *CT-RSA* (ABE, M., ed.), vol. 4377 of *LNCS*. Springer.

PAILLIER, P. & VERGNAUD, D. (2005). Discrete-Log Based Signatures May Not Be Equivalent to Discrete-Log. In: Roy [2005], pp. 1–20.

PAILLIER, P. & VILLAR, J. (2006). Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption. In: Lai & Chen [2006], pp. 252–266.

PAPADIMITRIOU, C. H. (1994). *Computational Complexity*. Addison Wesley.

PAULUS, S. & TAKAGI, T. (2000). A New Public-Key Cryptosystem over a Quadratic Order with Quadratic Decryption Time. *J. Cryptology* **13**(2), 263–272.

PEDERSEN, T. P. (1991). Distributed Provers with Applications to Undeniable Signatures. In: Davies [1991], pp. 221–242.

PEI, D., YUNG, M., LIN, D. & WU, C. (eds.) (2008). *Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Xining, China, August 31 - September 5, 2007, Revised Selected Papers*, vol. 4990 of *LNCS*. Springer.

POINTCHEVAL, D. (2001). Self-Scrambling Anonymizers. In: *Financial Cryptography, 4th International Conference, FC 2000* (FRANKEL, Y., ed.), vol. 1962 of *LNCS*. Springer.

POINTCHEVAL, D. & STERN, J. (2000). Security Arguments for Digital Signatures and Blind Signatures. *J. Cryptology* **13**(3), 361–396.

PRENEEL, B. (ed.) (2000). *Advances in Cryptology* - EUROCRYPT *2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, vol. 1807 of *LNCS*. Springer.

PRENEEL, B. (ed.) (2009). *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, vol. 5580 of *LNCS*. Springer.

RIVEST, R. L., SHAMIR, A. & ADLEMAN, L. M. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Comm. ACM* **21**, 120–126.

ROGAWAY, P. & SHRIMPTON, T. (2004). Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: *Fast Software Encryption, 11th International Workshop, FSE 2004* (ROY, B. K. & MEIER, W., eds.), vol. 3017 of *LNCS*. Springer.

ROY, B. (ed.) (2005). *Advances in Cryptology* - ASIACRYPT *2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Taj Coromandel, Chennai, India December 4-8, 2005, Proceedings*, vol. 3788 of *LNCS*. Springer.

SAKURAI, K. & YAMANE, Y. (1996). Blind Decoding, Blind Undeniable Signatures, and Their Applications to Privacy Protection. In: *Information hiding: first international workshop 1996* (ANDERSON, R., ed.). Springer.

SCHNORR, C. P. (1991). Efficient signature generation by smart cards. *J. Cryptology* **4**(3), 161–174.

SCHULDT, J. C. N. & MATSUURA, K. (2010). An Efficient Convertible Undeniable Signature Scheme with Delegatable Verification. In: ISPEC *2010* (KWAK, J., DENG, R. H., WON, Y. & WANG, G., eds.), vol. 6047 of *LNCS*. Springer. Full version available at the Cryptology ePrint Archive, Report 2009/454.

SEBERRY, J. & ZHENG, Y. (eds.) (1993). *Advances in Cryptology* - AUSCRYPT *'92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, vol. 718 of *LNCS*. Springer.

SHAHANDASHTI, S. F. & SAFAVI-NAINI, R. (2008). Construction of Universal Designated-Verifier Signatures and Identity-Based Signatures from Standard Signatures. In: *PKC 2008* (CRAMER, R., ed.), vol. 4939 of *LNCS*. Springer.

SHAMIR, A. (1985). Identity-Based Cryptosystems and Signature Schemes. In: *Advances in Cryptology* - CRYPTO*'84* (BLAKLEY, G. R. & CHAUM, D., eds.), vol. 196 of *LNCS*. Springer.

SHOR, P. W. (1994). Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: *FOCS*. IEEE Computer Society.

SHOUP, V. (1997). Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy [1997], pp. 256–266.

SHOUP, V. & GENNARO, R. (2002). Securing Threshold Cryptosystems against Chosen Ciphertext Attack. *J. Cryptology* **15**(2), 75–96.

STADJE, W. (2002). The Residues modulo m of Products of Random Integers. *Comb. Probab. Comput.* **11**(5), 529–540.

STERN, J. (ed.) (1999). *Advances in Cryptology* - EUROCRYPT*'99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, vol. 1592 of *LNCS*. Springer.

STERN, J., POINTCHEVAL, D., MALONE-LEE, J. & SMART, N. P. (2002). Flaws in applying proof methodologies to signature schemes. In: Yung [2002], pp. 93–110.

STINSON, D. (2006). *Cryptography: theory and practice*. Chapman & Hall/CRC. Taylor and Francis .

SUN, L. & LI, Y. (2005). XML undeniable signatures. In: *CIMCA/IAWTIC*. IEEE Computer Society.

THOMAS, T. & LAL, A. K. (2008). A Zero-knowledge Undeniable Signature Scheme in Non-abelian Group Setting. *I. J. Network Security* **6**(3), 265–269.

WANG, G., BAEK, J., WONG, D. S. & BAO, F. (2007). On the Generic and Efficient Constructions of Secure Designated Confirmer Signatures. In: *PKC 2007* (OKAMOTO, T. & WANG, X., eds.), vol. 4450 of *LNCS*. Springer.

WANG, G., QING, S., WANG, M. & ZHOU, Z. (2001). Threshold Undeniable RSA Signature Scheme. In: *ICICS* (QING, S., OKAMOTO, T. & ZHOU, J., eds.), vol. 2229 of *LNCS*. Springer.

WANG, G., ZHOU, J. & DENG, R. H. (2002). Cryptanalysis of the Lee-Hwang Group-Oriented Undeniable Signature Schemes. Cryptology ePrint Archive, Report 2002/150. `http://eprint.iacr.org/`.

WANG, X., LAI, X., FENG, D., CHEN, H. & YU, X. (2005). Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer [2005], pp. 1–18.

WANG, X. & YU, H. (2005). How to Break MD5 and Other Hash Functions. In: Cramer [2005], pp. 19–35.

WATERS, B. (2005). Efficient Identity-Based Encryption Without Random Oracles. In: Cramer [2005], pp. 114–127.

WIKSTRÖM, D. (2007). Designated Confirmer Signatures Revisited. In: TCC *2007* (VADHAN, S. P., ed.), vol. 4392 of *LNCS*. Springer.

WU, W., MU, Y., SUSILO, W. & HUANG, X. (2007a). Convertible Undeniable Proxy Signatures: Security Models and Efficient Construction. In: *WISA* (KIM, S., YUNG, M. & LEE, H.-W., eds.), vol. 4867 of *LNCS*. Springer.

WU, W., MU, Y., SUSILO, W. & HUANG, X. (2007b). Provably Secure Identity-Based Undeniable Signatures with Selective and Universal Convertibility. In: Pei *et al.* [2008], pp. 25–39.

YEUNG, W. K. Y. & HAN, S. (2003). Revocable Anonymity of Undeniable Signature Scheme. In: *IDEAL* (LIU, J., CHEUNG, Y.-M. & YIN, H., eds.), vol. 2690 of *LNCS*. Springer.

YUN, S.-H. & LEE, H.-W. (2004). The Undeniable Multi-signature Scheme Suitable for Joint Copyright Protection on Digital Contents. In: *PCM (3)* (AIZAWA, K., NAKAMURA, Y. & SATOH, S., eds.), vol. 3333 of *LNCS*. Springer.

YUN, S.-H. & LEE, H.-W. (2005). The Efficient Multipurpose Convertible Undeniable Signature Scheme. In: *KES (3)* (KHOSLA, R., HOWLETT, R. J. & JAIN, L. C., eds.), vol. 3683 of *LNCS*. Springer.

YUNG, M. (ed.) (2002). *Advances in Cryptology* - CRYPTO *2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, vol. 2442 of *LNCS*. Springer.

ZHANG, F., SAFAVI-NAINI, R. & SUSILO, W. (2004). An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In: Bao *et al.* [2004], pp. 277–290.

ZHANG, F., SAFAVI-NAINI, R. & SUSILO, W. (2005). Attack on Han et al.'s ID-based confirmer (undeniable) signature at ACM-EC'03. *Applied Mathematics and Computation* **170**(2), 1166–1169.

ZHANG, R., HANAOKA, G. & IMAI, H. (2007). Orthogonality between Key Privacy and Data Privacy, Revisited. In: Pei *et al.* [2008], pp. 313–327.

ZHU, H. (2004). Universal undeniable signatures. Cryptology ePrint Archive, Report 2004/005. `http://eprint.iacr.org/`.

List of Figures