Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science
Master's Program in Computer Science

Master's Thesis

Efficient Long-term Secure Universally Composable Commitments

submitted by

Oana-Mădălina Ciobotaru

on February 18, 2008

Supervisor

Prof. Dr. Michael Backes

Advisor

Dr. Dominique Unruh

Reviewers

Prof. Dr. Michael Backes

Dr. Dominique Unruh

Statement

Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Oana-Mădălina Ciobotaru
Saarbrücken, February 18, 2008

Declaration of Consent

Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Oana-Mădălina Ciobotaru
Saarbrücken, February 18, 2008

Acknowledgements

While carrying out this work, I learned that the most enjoyable and fruitful moments were the discussions with the people around me. Their professional enthusiasm taught me to pursue my work with passion and optimism.

I am grateful to my advisor, Dr. Dominique Unruh, for his ideas and continuous guiding. In many valuable discussions he made the topic accessible to me with patience and dedication.

I would like to thank Prof. Dr. Michael Backes for his support and energy and inspiring lectures in cryptography and advanced cryptography, and for offering me the opportunity to work on this topic.

For creating a great working environment, I would like to thank all members of Information Security and Cryptography group. Many thanks to Cătălin Hriţcu in particular for patiently proofreading drafts of my work.

Finally, I would like to thank my parents and my sister for always believing in me. Their continuous support was priceless and allowed me to be successful in the past two years.

Abstract

Long-term security ensures a protocol is secure during its execution while interacting with a computationally bounded adversary, as well as after its completion, when an attacker, trying to uncover the secrets used in the protocol, could become computationally unbounded. Universal composability guarantees the security of a cryptographic protocol even when it is arbitrarily composed with other protocols as well as instances of itself. The security notion which ensures preservation of long-term security under the strong universal composition property is called long-term universal composability. While this notion has been thoroughly defined and analyzed, the results so far can only prove the existence of long-term secure universally composable commitments, while completely overlooking the efficient design of such protocols for everyday use.

In this thesis we present a novel long-term universally composable secure commitment protocol that is both very efficient and plausible to use in practice. The efficiency of our scheme is due to the fact that we are able to modify, fine tune and combine three existing and efficient cryptographic primitives: a digital signature scheme by [CL02], the zero-knowledge proofs of knowledge by [CL02] and the commitment protocol by [DF02], while preserving the overall efficiency of the system. An additional challenge comes from the fact that our protocol needs to use a hardware device called a signature card, and for efficiency purposes, it should invoke the card as seldom as possible. We are able to reduce the number of necessary calls to the signature card to only one call.

Contents

Chapter 1

Introduction

1.1 Motivation

An important property of a cryptographic system is the degree of security it guarantees. However, in general, there is a trade off between different security levels and the aspects involving efficiency and practicality. These security guaranties are also subject to certain model assumptions describing the power that a possible adversary has when mounting an attack. The adversarial strength can vary from only computationally bounded power to unbounded power. The former case represents a realistic model for nowadays cryptographic systems and their adversaries, while the latter case gives a model of an all-powerful adversary.

A way to fill the gap between the notions mentioned above is by introducing the concept of long-term security. It considers the fact that during the execution of a protocol the adversary is computationally limited. Nevertheless, long-term security takes into account the following modeling assumption. At some point in time, but necessarily after the end of the protocol execution, due to advances in technology, the adversary could become computationally unbounded. One wants to design and prove the security of cryptographic systems with respect to such a model.

The intuitive description given above has taken into account only one instance of a protocol and its adversary in what is called a "stand alone" model. In many cases this does not suffice. From the security property described above, one cannot infer anything about the security of such a protocol when it runs within a bigger system or even when it is executed multiple times, in parallel or concurrently. On the other hand, good composability properties ensure that a secure protocol in the "stand alone" model remains secure independently of the way it interacts with other protocols or even with copies of itself. A general and also stringent composition property that captures this intuition is universal composability.

The two complementing notions of long-term security and universal composability are becoming more and more necessary requirements for today cryptographic systems. The first natural approach is to ensure the use of protocols or systems which fulfill each of these properties, individually. But this simple approach does not solve the problem. Indeed, Müller-Quade and Unruh have shown that the unconditionally hiding and UC secure commitment scheme described in [DN02], even though it is long-term secure in the "stand alone" model and also universally composable, does not maintain its long-term security property under composition. Thus, they have introduced the concept of long-term universal composability [MQU07], that encapsulates the two previously discussed notions and represents a solution for the impediment mentioned above.

The research done by Müller-Quade and Unruh in the area of long-term universal composability includes strong impossibility results together with descriptions of protocols for commitments

and zero-knowledge arguments that are secure in this model. Still, for these constructions, their approach is focused on proving the existence of commitment and zero-knowledge protocols that are secure in the long-term UC model under the assumption that one-way functions exist. This general cryptographic assumption does not allow for efficient protocol design.

1.2 Contribution

The current thesis goes one step further and investigates under which cryptographic and set-up assumptions the long-term UC commitment protocol can be made both more efficient to use and practical to build. We propose a new commitment scheme that is long-term UC secure if the strong RSA assumption holds. For this commitment, we build on the idea of using a signature card. Intuitively, this is a hardware device that signs only messages given to it by its owner, but never reveals the secret key used. This approach was first introduced in [MQU07]. From a practical point of view, using such a device represents a realistic set-up assumption as signature cards are already available on the market today.

Still, in the protocol proposed in [MQU07], the sender needs to query its signature card twice. First, it queries the card in the commit phase in order to obtain a signature on the pair of committed value and unveil message. Then it queries the signature card again in the unveil phase, in order to obtain a signature only on the unveil message. A query to the signature card is expensive for the protocol because in order to preserve the long-term UC property, each such query has to be followed by a zero-knowledge proof of knowledge of the signature obtained.

In our protocol, the second query is not needed: in the unveil phase the sender simply sends the recipient, in plain, the committed value together with the unveil message. In order to achieve this, we have slightly modified and used, without losing its initial security, the signature scheme from [CL02] and we gave it a trap-door property. We rely on this property for building the simulator required in long-term UC definition. Moreover, this signature scheme also allows us to use very efficient techniques [CL02] for implementing zero-knowledge proofs of knowledge of signatures. The resulting protocol is a six-round long-term UC secure commitment that requires only thirteen exponentiations from the sender and eleven exponentiations from the recipient. These exponentiations are done modulo a special RSA modulus n.

1.3 Overview

This thesis is structured as follows. The basic definitions and concepts used in this work are included in chapter 2. Then, in chapter 3, we describe the universal composability model and review the most important results obtained in this area. The related but more stringent notion of long-term universal composability is defined and analysed in chapter 4. There we also include a short description of the already existing results and we mention the protocols that have been proposed so far. Our contribution, mainly the efficient long-term UC commitment protocol that we construct, is described in chapter 5. There we also include the proof of its security. For more clarity, the descriptions of the primitives used have also been added. Finally, in chapter 6 we give an overview of this work and mention further related problems that can be solved using the techniques developed here.

Chapter 2

Basic Cryptographic Notions

Oded Goldreich states that "modern cryptography is based on the gap between efficient algorithms provided for the legitimate users and the computational infeasibility of breaking these algorithms via illegitimate adversarial actions ". In general, the requirement that a protocol is secure for all possible adversaries is quite strong and hard to fulfill. For example, a lucky adversary who just makes a correct guess of the secret of a legitimate user can impersonate himself as one. Consequently, in cryptography such a requirement is replaced with a weaker one. In the case of our example, the requirement is that impersonating oneself as another user should happen only with very small probability. This implies also that the event of guessing the secret should occur very rarely. Our first definition formalizes the notion of very small probability.

The secret, known only by the user, and which is usually a bit string, is intuitively harder to guess if its length increases. This length is parameterized by a *security parameter* k which is a natural number. All machines, adversaries, protocols or cryptographic tools have their own security parameter. We can now express probabilities for certain security related events in terms of this parameter k. Intuitively, an event occurs with very small probability if its probability approaches zero faster than any polynomial. If a function has this property, then it is called *negligible*.

Definition 1 (Negligible Functions)**.** *We call a function $\mu : \mathbb{N} \to \mathbb{R}_0^+$ negligible if for every positive polynomial $p(\cdot)$ there exists an $n_p \in \mathbb{N}$ such that for all $n > n_p$*

$$\mu(n) < \frac{1}{p(n)}.$$

The concept of non-negligible functions simply denotes all functions for which the above definition is not fulfilled. In order to avoid possible confusions, we explicitly define this concept next.

Definition 2 (Non-Negligible Functions)**.** *We call a function $\mu : \mathbb{N} \to \mathbb{R}_0^+$ non-negligible if there exists a positive polynomial $p(\cdot)$ such that for every $n_p \in \mathbb{N}$ there exist an $n > n_p$ such that*

$$\mu(n) \geq \frac{1}{p(n)}.$$

Related to the above definitions, we have the concept of *overwhelming* probability. Intuitively, the probability for an event to occur is overwhelming if the complement probability, i.e. the probability that the event does not occur, is negligible. In order to preserve the full generality, we describe this property also in terms of functions, in general.

Definition 3 (Overwhelming Functions)**.** *We call a function $f : \mathbb{N} \to \mathbb{R}_0^+$ overwhelming if for every positive polynomial $p(\cdot)$ there exist an $n_p \in \mathbb{N}$ such that for all $n > n_p$*

$$f(n) > 1 - \frac{1}{p(n)}.$$

It is clear that if an adversary is allowed to try all the possible values in order to guess a secret, then it will be able to find it. Thus, we have to restrict the power of an adversary to probabilistic polynomial-time. An adversary or algorithm verifying this property is called *efficient*.

Definition 4 (Efficient Algorithms)**.** *Let k be the security parameter. An algorithm is called efficient iff it is computable in probabilistic polynomial-time (PPT) in k.*

In the following we are going to use the notation "$\leftarrow$" when referring to the assignment made by a probabilistic algorithm. If a value is chosen according to the random distribution on a certain domain, we denote this by "$\xleftarrow{r}$". A statement of the form:

$$Pr(b(x_n) : \{x_i \leftarrow A_i(y_i)\}_{1 \leq i \leq n})$$

means that the probability that $b(x_n)$ is true after the value x_n was obtained by running algorithms $A_1, \ldots, A_n$ on inputs $y_1, \ldots, y_n$, is α.

If for a certain type of problem there is no (known) efficient algorithm that can solve it, for a random given instance of it, we call that problem hard or intractable. Intuitively, modern cryptography relies on the existence of problems that are hard to solve. One of the most common and general assumptions is the existence of one-way functions. Loosely speaking, these functions are deterministically computable by a polynomial time algorithm, but infeasible to invert in probabilistic polynomial-time.

Definition 5 (One-way Functions)**.** *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called one-way iff it fulfills the following conditions:*

- *There exists a deterministic polynomial-time algorithm that on input x outputs $f(x)$.*
- *For every efficient algorithm A the following holds: Let $x \xleftarrow{r} \{0,1\}^k$ and $x' \leftarrow A(f(x), 1^k)$, then the probability $Pr(f(x) = f(x'))$ is negligible.*

Even though the existence of one-way functions is still an open problem, many cryptographic constructions build upon the assumption that they exist. For example, it is known that secure digital signature schemes exist if and only if one-way functions exist. Intuitively, a digital signature is the cryptographic corespondent for a hand-written signature. It is used to ensure that the signed message comes from a certain party. It also provides that an adversary cannot "forge" such a signature without being detected.

Definition 6 (Public-Key Digital Signature Schemes)**.** *A signature scheme (Gen, Sig, Ver) is a triple of probabilistic polynomial-time algorithms fulfilling the following conditions:*

- *On input 1^n, Gen outputs a pair of bit-strings (pk, sk) called the public key and the secret key, respectively.*
- *For every pair (pk, sk) in the range of Gen and for every $x \in \{0,1\}^*$, the algorithms Sig and Ver satisfy*

$$Pr(\, Ver(pk, x, Sig(sk, x)) = 1) = 1.$$

The algorithm Sig is called the signing algorithm and the algorithm Ver is called the verification algorithm.

From the security point of view, there is an entire range of attacks against such a digital signature scheme. For the purpose of this work we require a digital signature scheme such that there is no efficient adversary which only by knowing the public key can produce a new valid (message, signature) pair with more than negligible probability. This should hold even after the adversary obtains polynomially many signatures on messages of its choice from the signature scheme. We underline the fact that not only the pair (message, signature) has to differ from all the pairs queried so far, but also the message forged by the adversary has to be a new one. A scheme that meets these requirements is said to be secure against existentially-unforgeable adaptively-chosen message attacks.

Definition 7 (EF-CMA Security for Digital Signature Schemes)**.** *The triple* (Gen, Sig, Ver) *representing a digital signature scheme is existentially-unforgeable under an adaptively-chosen message attack if for every probabilistic polynomial-time algorithm* A *there is a negligible function* $\mu()$ *such that*

$$Pr(m \neq m_i \text{ for } i = 1 \dots n \wedge Ver(pk, m, \sigma) = 1 : (pk, sk) \leftarrow Gen(1^k), m_i \leftarrow A(pk, m_1, \sigma_1, \dots, m_{i-1}, \sigma_{i-1}), \sigma_i \leftarrow Sig(sk, m_i) \text{ for } i = 1 \dots n, (m, \sigma) \leftarrow A(pk, m_1, \sigma_1, \dots, m_n, \sigma_n)) \leq \mu(k). \quad (2.1)$$

Next we describe a cryptographic building block called *commitment*. Since in the next chapters we give the exact definition of the commitment we want to implement (mainly the ideal functionality for commitment) we only present here the intuition for such a scheme. In a commitment protocol two parties are involved: the sender and the recipient. Loosely speaking, the commitment acts as an envelope. First, the sender commits to its value, which intuitively is like placing the value in an envelope and sealing it. Then, the sender unveils its value to the recipient in the same way as opening an envelope, and the recipient accepts if it is convinced that this was the value in the first place. The correctness of such a protocol means that if both parties act according to the protocol, then the recipient should be convinced in the end, with overwhelming probability.

The biding property intuitively ensures that not even a cheating sender can open the commitment with more than a negligible probability to a different value from the one it had initially committed to. Depending on the computational power of the cheating sender, a commitment scheme can be perfectly binding or computationally binding. If the commitment scheme is secure against even all-powerful cheating senders it is called perfectly binding, while being secure only against efficient senders results in a computationally binding commitment scheme.

The property that protects the sender against malicious recipients, that may try to open the commitment before the unveil phase, is called the hiding property. As in the description made above, a commitment can be computationally hiding or perfectly hiding. One would like to have a commitment scheme that is both perfectly hiding and perfectly binding. It has been shown though that such a scheme does not exist, but one can require at most a perfectly binding and computationally hiding commitment or perfectly hiding and computationally binding one. For further discussions on commitments, we refer to [Gol01]

Next we describe the notion of indistinguishability of probability distributions, depending on the computational power of the distinguisher. First, we have the case of probabilistic polynomial-time distinguishers.

Definition 8 (Computational Indistinguishability)**.** *Let* $\{X_n\}_{n\in\mathbb{N}}$ *and* $\{Y_n\}_{n\in\mathbb{N}}$ *be two probability ensembles indexed by* $\mathbb{N}$*, i.e. each* X_i *and* Y_i*, respectively, is a random variable.* $\{X_n\}_{n\in\mathbb{N}}$ *and*

$\{Y_n\}_{n\in\mathbb{N}}$ are computationally indistinguishable if for every probabilistic polynomial-time algorithm D,

$$|Pr(D(1^n, \alpha) = 1 : \alpha \leftarrow X_n) - Pr(D(1^n, \alpha) = 1 : \alpha \leftarrow Y_n)|$$

is negligible in n.

If in the previous definition one allows the distinguisher D to be computationally unlimited, we obtain the notion of statistical indistinguishability.

Definition 9 (Statistical Indistinguishability)**.** *Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be two probability ensembles. We call $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ statistically indistinguishable if for every algorithm D, possibly even computationally unbounded,*

$$|Pr(D(1^n, \alpha) = 1 : \alpha \leftarrow X_n) - Pr(D(1^n, \alpha) = 1 : \alpha \leftarrow Y_n)|$$

is negligible in n.

In this thesis, the protocols we use require working in the group of invertible elements modulo n, which is denoted by $\mathbb{Z}_n^*$. In general, cryptographic protocols using computations mod n, need n to be either a prime or an RSA modulus. The latter notion is formalized next.

Definition 10 (RSA modulus)**.** *A $2k$-bit number n is called an RSA modulus if $n = p \cdot q$, where p and q are k-bit primes.*

For any integer n, the cardinality of the group $\mathbb{Z}_n^*$ is measured by the Euler's totient function $\phi(n)$. In the case of an RSA modulus n, we have $\phi(n) = (p-1)\cdot(q-1)$. We say that an RSA modulus is chosen uniformly at random with security parameter k if p and q are random k-bit primes. Next, we define a particular type of RSA modulus.

Definition 11 (Special RSA modulus)**.** *An RSA modulus is special if $p = 2\cdot p' + 1$ and $q = 2\cdot q' + 1$, where p' and q' are also prime numbers.*

In spite of the extensive research directed towards the construction of efficient integer factoring algorithms, the best known algorithms for this purpose have sub-exponential running time. In particular, for factoring a random special RSA modulus, a more efficient algorithm then the best general factoring algorithm is not known. Thus, the intractability of factoring a random special RSA modulus represents a reasonable assumption. In our work we will use computations modulo a special RSA modulus n, hence the remaining part of this chapter is related to this notion.

Definition 12 (Factoring Assumption for Special RSA Moduli)**.** *Let n be a $2k$-bit special RSA modulus. Then there is no efficient algorithm A that outputs the factorization of n with more than negligible probability.*

Note that even a more general factoring problem is considered hard, namely factoring a randomly chosen RSA modulus. A known implication of the factoring assumption for a random RSA modulus n is that for such an n it is hard to compute the value $\phi(n)$. Another cryptographic assumption, motivated by the existing technology and algorithms, is the strong RSA assumption for a special RSA modulus: given an RSA public key (n, e), where n is a special RSA modulus, and given a random element $u \in \mathbb{Z}_n^*$, it is conjectured hard to compute values $e > 1$ and v such that $v^e = u \bmod n$. Let *sRSAmodulus* denote the algorithm that, on input 1^k outputs a random special RSA modulus with security parameter k. Then the assumption can be stated more formally:

Definition 13 (Strong RSA Assumption for Special RSA Moduli)**.** *We assume that for all polynomial-time circuit families* $\{A_k\}_{k\in\mathbb{N}}$*, there exists a negligible function* $\mu(k)$ *such that*

$$Pr(e > 1 \wedge v^e = u \ mod\ n : n \leftarrow sRSAmodulus(1^k), u \leftarrow \mathbb{Z}_n^*, (v, e) \leftarrow A_k(n, u)) \leq \mu(k).$$

The pair (n, u) generated above is called a *general instance* of the *flexible RSA problem*. By $QR_n \subseteq \mathbb{Z}_n^*$ we will denote the set of quadratic residues modulo n, i.e. the elements $a \in \mathbb{Z}_n^*$ suct there exists $b \in \mathbb{Z}_n^*$ such that $b^2 = a \bmod n$. Note that since one quarter of all elements of $\mathbb{Z}_n^*$ are quadratic residues, the strong RSA assumption implies that it is hard to solve *quadratic* instances of the flexible RSA problem. In the sequel, by *instance* of the flexible RSA problem, we will mean a *quadratic* rather than general instance.

At this point, it is important to note the following implication. If the strong RSA assumption holds, then factoring a randomly chosen special RSA modulus is hard. Indeed, arguing in an informal way, we can assume by contradiction that the strong RSA assumption holds but factoring a randomly chosen RSA modulus is not hard. This last fact implies that there exists an algorithm A that can factor a randomly chosen special RSA modulus with non-negligible probability. Also, it is known the fact that computing the square roots of a quadratic residue modulo $n = p \cdot q$, where p and q are primes, is possible in polynomial time if the factorization of n is revealed. By combining the last two statements, we can construct in a clear way an algorithm B that breaks the strong RSA assumption with non-negligible probability. Thus, we obtain a contradiction and the implication we made is proved.

If n is a special RSA modulus, then it is known, for example from [CL02], that the group QR_n is cyclic, i.e. there exist at least one generator for it.

Lemma 14 (QR_n is Cyclic for a Special RSA Modulus n)**.** *If* $n = p \cdot q$*,* $p = 2 \cdot p' + 1$*,* $q = 2 \cdot q' + 1$ *is a special RSA modulus, then* QR_n *is a cyclic group under multiplication, of size* $p' \cdot q'$*, where all but* $p' + q'$ *of the elements are generators.*

Chapter 3

Universal Composability

3.1 Intuition

Proving in a rigorous manner that a protocol completes its task in a secure way is an important part of the design of cryptographic protocols. In order to tackle the security issues, an appropriate model has to be chosen and within this model, a definition for the security of the task at hand has to be constructed. Then it has to be shown, using a mathematical proof, that the designed protocol fulfils that definition.

If we follow this course of action with a model that, for example, does not take into consideration certain properties that a possible adversary could have, then we restrict from the beginning the possible situations where the protocol could be used securely. This is why, the definition should guarantee security with respect to any possible adversarial behavior.

Finding proper models and good security definitions for cryptographic tasks it is, in general, both necessary and a nontrivial endeavor. An important fact that has to be taken into consideration is that the protocol should maintain a certain level of security not only when it runs alone and it interacts just with its possible adversary, in what it is called the stand-alone model. In many real-world applications the same level of security has to be met even when the protocol is interacting with different copies of itself or it runs within more complex systems. The first approaches used for defining security notions like zero-knowledge [GO94, GMR89] have taken into consideration only the above mentioned stand-alone model. Later, it has been shown that there exist protocols fulfilling such definitions, but they are not closed under parallel or concurrent composition.

In the following we will summarize an example by [Fei92] of a zero-knowledge protocol in the stand-alone model which fails to preserve this property even when only two of its instances run concurrently. Let π be a zero-knowledge proof for a relation R. This means that there exist a prover P that transmits a value x to a verifier V and additionally wants to convince V that he has a secret witness w such that $R(x, w)$ holds. Since the protocol is zero-knowledge, V learns nothing more from P, besides the fact that V has a witness. The example also assumes a "puzzle system" where both the prover and the verifier can generate puzzles. While the prover can solve any puzzle, not only the ones he generates, the verifier can solve puzzles only with negligible probability and additionally he can not even distinguish between a correct solution to a puzzle and a random string. Such a "puzzle system" is shown to exist [Fei92], if for example the prover P holds some trapdoor information. Now, let π' the protocol where the parties run π in the beginning. Then P sends a random puzzle p to V and the verifier responds with a pair (s_1, q), where q is also a puzzle. If the value s_1 represents a solution for the puzzle p, then P responds

with the secret witness w, else P sends to V a solution s_2 for the puzzle q.

First, we argue in an informal manner that the protocol π' is zero-knowledge in the stand alone model. Indeed, since V cannot solve puzzles, then P will send the witness w to V only with negligible probability. Moreover, the fact that V receives a solution for q is not a problem in the stand-alone model, since V can not distinguish a solution from a random string. On the other hand, assume the situation where two instances of the protocol are running concurrently, with the same input (x, w) for P. A corrupted verifier V' could wait to receive from P the puzzles p and p' from both instances of π'. Then V' sends (s, p') to the first instance of π', where s is an arbitrary string. In response, V' gets a solution s' which he sends to the second instance of π'. Thus, V' receives the witness w from P in the second instance of π' as he has sent a correct solution for p'.

From this example, it is clear that a good security definition should take into consideration not only the protocol itself and its adversary, but also other possible protocols or even copies of the initial protocol. In order to formalize the intuition of the exterior world with respect to a given protocol π, the notion of environment was introduced. Intuitively, the environment for a certain protocol contains all the other protocols, systems or users, together with their own adversaries, that may or may not interact with π. It is important to note that the adversary for protocol π is not considered to be a part of the environment.

In this context, one way to include certain concerns regarding possible unwanted interactions between an environment and the specific protocol π is to represent that specific environment within the security definition. Such an approach was taken, for example, regarding the definition of concurrent zero-knowledge in [DNS98]. In this way, the complexity of the security definition depends on the complexity of the environment. A clear drawback is that different security definitions have to be used for different environments and concerns.

Another approach is to use strong security definitions that treat protocols in the stand-alone model, but can guarantee secure composition with other protocols and even with themselves.

Throughout this work, we follow the definitional approach introduced in [Can01], and further developed in [Can05]. In this definition, a single instance of the protocol π is inspected and if the definition is verified we say that the protocol is *universally composable*. First we start with an intuitive overview of the framework, and later in the chapter we describe all necessary details and consequences. In order to determine whether a given protocol securely implements a given task, first we define the ideal process for caring out that task. Intuitively, in an ideal process for a given task, all parties give their inputs directly to a *trusted party*, which in the following is also called the *ideal functionality for that task*. This ideal functionality can be regarded as a "formal specification" of the security requirement of the task. According to the security definition given in this framework, a protocol securely implements a task if any damage that can be caused by an adversary while interacting with the protocol, can also be caused by an adversary interacting with the ideal process for that task. We say that the protocol runs in a real-world model and the ideal functionality runs in the ideal-world model.

3.2 Real-world Protocols

More formally, let π be a cryptographic protocol. The real-world model for the execution of protocol π contains the following Turing machines: an interactive Turing machine (ITM) $\mathcal{Z}$ called the environment, a set of ITMs representing the parties running the protocol π and an ITM representing the adversary. We now have a more detailed look at each of these ITMs.

The environment $\mathcal{Z}$, as mentioned above, represents everything that is external to the current execution of π and it is modeled as an ITM with auxiliary input. The environment can provide,

throughout the course of the protocol execution, inputs to parties running π as well as to the adversary. These inputs are a part of the auxiliary input that $\mathcal{Z}$ has received itself. In case inputs were sent, then $\mathcal{Z}$ receives all the outputs that are generated by the parties and the adversary. It is important to note that the only interaction between the environment $\mathcal{Z}$ and the parties is when the environment sends the inputs and receives the outputs. This can happen at the beginning of the protocol, as well as during the protocol execution. Finally, at the end of π, the environment outputs all the messages received.

The adversary, as explained above, can receive one or more input messages from $\mathcal{Z}$ at any moment during the protocol execution and it can send in the same manner its output messages back to $\mathcal{Z}$. Since we want to capture any possible adversarial behaviour, the communication between the adversary, denoted by $\mathcal{A}$, and $\mathcal{Z}$ is not limited only to this. The two ITMs can communicate freely throughout the course of the protocol and they can exchange information after any message sent between the parties and after any output a party would make.

Next, we need to look closer at the notion of corruption. By considering a party P corrupted we mean that from that point on that adversary has access to all the input and communication messages send or received by that party, and depending on the communication model $\mathcal{A}$ can decide to alter such messages in any way it wants. Moreover, all the past incoming or outgoing messages of P are known to $\mathcal{A}$.

In order for $\mathcal{A}$ to corrupt a party P, it first has to inform $\mathcal{Z}$ by sending it a corruption message $(corrupt, P)$. Thus $\mathcal{Z}$ is aware at any given moment about the corruption state of all parties. There are two corruption models depending on the moment when the adversary $\mathcal{A}$ can corrupt a party. The adversarial behaviour model which allows $\mathcal{A}$ to corrupt parties only in the beginning of the protocol, before the respective parties receive their inputs from $\mathcal{Z}$, it is called the static model of corruption. On the contrary, if $\mathcal{A}$ is allowed to corrupt a party at any given moment during the execution of π, then the respective model is called adaptive. Another way to look at the corruption model is by inspecting whether the adversary is passive, and only learns all inputs and communication messages a corrupted party sends and receives, or if $\mathcal{A}$ is active. This second case would imply that $\mathcal{A}$ is allowed to modify any input a corrupted party P gets and he could also modify all the communication messages.

Throughout this work, we are using the static and active corruption model. In order simplify the presentation of the proofs, we use an equivalent definition for the static model of corruption. As mentioned above, a static adversary can choose whether to corrupt a party or not, only in the beginning of the protocol. Hence, by having fixed the moment of corruption, we can skip sending and receiving the corruption messages. Instead, we assume that the corrupted party or parties are fixed from the start and the adversary does not have to choose them. It is easy to see that the previous definition of static adversary is equivalent to the latter formulation, which will be used in this work.

After the execution of the protocol has ended, the environment $\mathcal{Z}$ outputs its view of that execution. This view contains the messages that $\mathcal{Z}$ has received from the adversary $\mathcal{A}$ and the output messages of all parties. Formally, let $EXEC_{\pi,\mathcal{A},\mathcal{Z}}\ (k, z)$ denote the output of $\mathcal{Z}$ in an execution of the protocol π with adversary $\mathcal{A}$ and environment $\mathcal{Z}$, where k is the security parameter and z is the auxiliary input to the environment $\mathcal{Z}$. We denote by $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$ the family of random variables $\{EXEC_{\pi,\mathcal{A},\mathcal{Z}}\ (k, z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$.

The adversarial behavior in the case of a corrupted party was discussed above, but we need to describe also the possible interference of the adversary with the communication between honest parties. The most basic model for the UC framework ensures that all messages are handed to the adversary and the adversary delivers messages of its choice to all parties. This model makes no assumption on the properties of the communication such as the authenticity, secrecy or synchrony of the messages delivered. Even though this is a very simple model, it is very hard to work with

it for constructing useful protocols. For the more specialised models of authenticated, secure or synchronous communication, an ideal functionality is added to the basic model to capture the respective properties.

Authenticated communication assumes that even though the adversary could have access to the content of messages, it cannot alter them without being detected. The property that messages are all delivered and without delay from the moment they were generated is captured by the synchronous communication model. The secure communication model has an even more stringent definition than the authenticated communication notion. It assumes the adversary receives all messages, but it has neither access to the content of communication, nor possibility to modify any message without breaking the authenticity of that message. Thus, in this model, the adversarial capabilities are limited to either delaying or not delivering some or all messages between the uncorrupted parties.

3.3 Ideal Process and Ideal Functionalities

In order to formalize the ideal process, we do not want to define a different model from that formulated above, but we rather need to adapt it. In the same way as in the real-world, the environment $\mathcal{Z}$ is the only ITM that can send input, at any moment, to the ideal process parties and the ideal adversary. In the case of the ideal process, the adversary is called the ideal simulator and is denoted by $\mathcal{S}$. Moreover, $\mathcal{Z}$ receives all the outputs generated by the parties he has direct access to, as well the possible outputs of $\mathcal{S}$.

The first difference is that in the ideal model there exists a trusted party that cannot be directly accessed by the environment. This trusted party, also called ideal functionality, works as follows. All the parties involved in the ideal process give their input directly to the ideal functionality. On its turn, the ideal functionality computes an output for each of the parties. Such an output message could also be an empty message. These messages are sent directly to the ideal parties. Thus, it is clear that the role of the ideal functionality is to securely receive messages, perform computations and output results to the other ideal parties. Additionally, it is important to note that environment $\mathcal{Z}$ has direct access to all these ideal parties. As they do not take an active role in the computation and just send input and receive output to and from the ideal functionality, they are called dummy parties of the ideal functionality.

The second difference from the real-world model concerns the behavior of the dummy parties. Messages delivered by the adversary to these parties are ignored. The intention is that in the ideal protocol the adversary should send corruption messages directly to the ideal functionality. The ideal functionality will then determine the effect of corrupting a party. A typical response is to let the adversary know all the inputs received and outputs made by the party so far.

The environment $\mathcal{Z}$ and the simulator $\mathcal{S}$ can communicate freely during the execution of the ideal process. As mentioned in the section 3.1, in order to prove the security of a given protocol π, for every possible adversary attacking π in the real world we have to construct an ideal simulator, in the ideal world. This simulator's interaction with $\mathcal{Z}$ and the ideal functionality has to intuitively "imitate" the interaction between π, $\mathcal{Z}$ and the real adversary $\mathcal{A}$. As we have already seen, adversary $\mathcal{A}$ controls the communication among the parties performing the real world-protocol. More precisely, $\mathcal{A}$ is allowed to delay messages or even decide not to send some or all messages that represent the communication among real-world protocol parties. In order to model the same behavior for $\mathcal{S}$, every time the ideal functionality wants to output a message, it first informs the simulator. If the simulator agrees, then the ideal functionality can make the respective output.

Similar to the real-world model, the environment $\mathcal{Z}$ outputs its view in the end of the ideal

process execution. The view contains all the messages received from the simulator as well as all the messages that the dummy parties output to $\mathcal{Z}$. More formally, by $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)$ we denote the output of $\mathcal{Z}$ in an execution of the ideal process with the trusted party $\mathcal{F}$, simulator $\mathcal{S}$ and environment $\mathcal{Z}$, where k is the security parameter and z is the auxiliary input to the environment $\mathcal{Z}$. We denote by $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ the family of random variables $\{EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$.

3.4 Protocol Emulation and the UC Theorem

Having defined all the details related to the protocol execution, both in the real and the ideal world, we can now proceed to formalize the notion of *emulation*. First, we define what it means that a real-world protocol π *emulates* another real-world protocol ϕ. We want to model the fact that "any damage" which can be done by an adversary interacting with π, can also be produced by some adversary interacting with ϕ. The environment $\mathcal{Z}$ is the ITM deciding whether the "effects" of the interaction between each protocol and its respective adversary are "almost identical" and this indistinguishability property should hold for every computationally bounded $\mathcal{Z}$.

All the ITMs used in either of the protocol executions for π or ϕ, including the environment $\mathcal{Z}$, are computationally bounded. Thus, in order to guarantee that $\mathcal{Z}$ cannot decide with which protocol and adversary it is interacting, it is sufficient if we formalize the notion of emulation in terms of computational indistinguishability. The environment $\mathcal{Z}$ will act as a distinguisher for the two protocol executions. Since all the information $\mathcal{Z}$ gains throughout its interaction is contained within the view $\mathcal{Z}$ outputs in the end, it is sufficient to compare the two views. Essentially, protocol π emulates protocol ϕ if for every adversary $\mathcal{A}$ there is an ideal simulator $\mathcal{S}$ such that for every environment $\mathcal{Z}$ the views of the two interactions are computationally indistinguishable.

Definition 15 (Protocol Emulation)**.** *Let π and ϕ be PPT protocols. We say that π UC emulates ϕ if for every PPT adversary $\mathcal{A}$ there is a PPT simulator $\mathcal{S}$ such that for every PPT $\mathcal{Z}$ the two families of random variables $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$ and $EXEC_{\phi,\mathcal{S},\mathcal{Z}}$ are computationally indistinguishable.*

This general notion of emulation can be adapted to the special case of the ideal process. We say that a protocol realizes an ideal functionality if it emulates the ideal process for that functionality.

Definition 16 (Realizing an Ideal Functionality)**.** *Let $\mathcal{F}$ be an ideal functionality and let π be a protocol. We say that π UC realizes $\mathcal{F}$ if π UC emulates the ideal process for $\mathcal{F}$.*

Intuitively, the above definition helps us transfer the security implied by an ideal functionality into a real-world system. More precisely, once we have proved that a real world protocol π realizes an ideal functionality $\mathcal{F}$, in other words, π is as secure as $\mathcal{F}$, we would like to use protocol π within any system that requires one of it components to have the properties defined by $\mathcal{F}$. In order to do this, we need to extend the existing model for protocol execution to a new type of protocol that combines properties of both real and ideal world. Such a protocol will be called hybrid and will allow its parties to access or make calls to instances of ideal functionalities.

Thus, in addition to communicating through the adversary in the usual manner in a real world-protocol, any party or set of parties of a hybrid protocol can invoke an ideal functionality by calling the corresponding ideal process for that functionality. More precisely, we call π an $\mathcal{F}$-hybrid protocol if π includes calls to the ideal process for $\mathcal{F}$. This means that a party or set of parties will invoke the dummy parties for $\mathcal{F}$, which in turn will invoke $\mathcal{F}$. Upon corruption messages received by the parties of π, the behaviour is that specified in the real-world. If the dummy parties of $\mathcal{F}$ are sent corruption messages, then $\mathcal{F}$ acts as described in the ideal model. If a protocol makes use of multiple different ideal functionalities, the hybrid model can be extended in a natural way.

Next, we define a composition operation that will be stated as usual, in terms of real-world protocols. We can look then into the special case of the ideal process and the ideal functionalities. Let π be a protocol that makes calls to a protocol ϕ and let ρ be a protocol that UC emulates protocol ϕ. The composed protocol $\pi^{\rho/\phi}$ is identical to the protocol π with the following modifications.

- Each time π has to pass an input to an ITM running protocol ϕ, then $\pi^{\rho/\phi}$ will pass this input to an ITM running ρ.
- Each time $\pi^{\rho/\phi}$ receives an output passed from an ITM P running the protocol ρ, it proceeds as π when receiving the same output from P.

This operation is called universal composition of protocols π, ϕ and ρ. In particular, when the protocol ϕ is the ideal process for some ideal functionality $\mathcal{F}$, the composed protocol is denoted by $\pi^{\rho/\mathcal{F}}$. We note that if π, ϕ and ρ are PPT protocols, then $\pi^{\rho/\phi}$ is also a PPT protocol.

The universal composition operation was used in [Can05] to construct the following result that represents the core of the universal composability research. It can be used as a tool for modular design and analysis for complex protocols. Loosely speaking, the universal composition theorem allows one to prove that the overall security of a system stays unchanged if calls to a protocol ρ are replaced by calls to protocol ϕ, when ρ UC emulates ϕ.

Theorem 17 (Universal Composition Theorem)**.** *Let π, ρ, ϕ be probabilistic polynomial time multi-party protocols such that ρ UC emulates ϕ and π makes calls to protocol ϕ. Then protocol $\pi^{\rho/\phi}$ UC emulates protocol π.*

The following corollary states the conditions under which a complex system UC realizes an ideal functionality.

Theorem 18 (Realizing Functionalities by Universal Composition)**.** *Let $\mathcal{F}$, $\mathcal{G}$ be ideal functionalities such that $\mathcal{F}$ is PPT. Let π be a protocol that UC realizes $\mathcal{G}$ and makes calls to ideal functionality $\mathcal{F}$. If ρ is a protocol that securely realizes $\mathcal{F}$, then the composed protocol $\pi^{\rho/\mathcal{F}}$ UC realizes functionality $\mathcal{G}$.*

3.5 Defining and Realizing some UC Tasks

The framework detailed above allows us to define in a general form a set of ideal functionalities that are commonly used for proving security of cryptographic tasks. Different assumptions are presented later in this section, which make possible the construction of secure UC protocols fulfilling such definitions. The short overview ending the section contains some important results published so far in the UC model.

We start by investigating the common reference string and the coin toss functionalities. These are two simple functionalities that model the concept of randomness shared by two or more parties.

Definition 19 (Common Reference String Functionality $\mathcal{F}_{CRS}^{\mathcal{D}}$)**.** *The $\mathcal{F}_{CRS}^{\mathcal{D}}$ chooses for its first activation a value r according to some distribution $\mathcal{D}$. Upon any input from a party P_i, it sends r to the adversary $\mathcal{A}$. When the adversary accepts the delivery, $\mathcal{F}_{CRS}^{\mathcal{D}}$ sends r to P_i. (It is important to note that all parties get the same value r).*

Definition 20 (Coin Toss Functionality $\mathcal{F}_{CT}$)**.** *When both parties have given some input, $\mathcal{F}_{CT}$ chooses a random value r in a domain M and sends it to $\mathcal{A}$. Upon receiving the acceptance of delivery from the adversary, it sends r to both parties.*

The two functionalities specified above look identical if some details in the definitions are overlooked. Nevertheless, the $\mathcal{F}_{CRS}^{\mathcal{D}}$ functionality computes and stores the value r before any input is given to it, while the $\mathcal{F}_{CT}$ functionality chooses the random value only after all the parties have sent their input.

Zero-knowledge proofs and commitments represent frequent building blocks for many larger cryptographic systems. The respective ideal functionalities are thus defined next.

Definition 21 (Zero-Knowledge Functionality $\mathcal{F}_{ZK}^{R}$)**.** *Let R be an NP relation and let $\mathcal{P}$ and $\mathcal{V}$ be two parties. The functionality $\mathcal{F}_{ZK}^{R}$ behaves as follows. Upon the first input of (x, w) from $\mathcal{P}$ satisfying xRw, it sends x to $\mathcal{V}$ as soon as the adversary has accepted the delivery.*

Definition 22 (Commitment Functionality $\mathcal{F}_{COM}^{C,R}$)**.** *Let C and R be two parties. The ideal functionality $\mathcal{F}_{COM}^{C,R}$ behaves as follows. Upon the first input (commit,x) from C it sends (committed) to R. When R receives the message (committed), it outputs (accept) to the environment $\mathcal{Z}$. Upon input (unveil) from C, $\mathcal{F}_{COM}^{C,R}$ sends (x) to R. When R receives the message x, it outputs (accept,x) to the environment.*

We call C the sender and R the recipient.

The notions mentioned above represent examples of common cryptographic tasks that can be defined in the UC framework. Next, we analyze to what extent such definitions are realizable and under which assumptions. In [Can01] it was shown, with a sketch proof, that if at most one third of the parties are corrupted, then any ideal functionality is UC realizable assuming synchronous communication channels and the existence of trapdoor permutations. Even if this is a clear positive result, having two thirds of the parties honest may be hard to achieve in many real-life situations, thus the problem was further investigated. In the case of an honest minority, it becomes important whether the functionality to be realised is a two-party or a multi-party functionality (i.e. when more than two parties give and receive input).

There exist strong impossibility results for realizing two-party functionalities, when at least one of the parties is corrupted and the protocol has very basic set-up assumptions. For example, in [CF01] it is shown that the ideal commitment can not be realised by any protocol, when at most authenticated communication is provided. This was extended to a series of impossibility results mainly in [CKL03, BDD+06]. Thus, so far it is known that almost all non-trivial deterministic and many probabilistic functionalities, including zero-knowledge and coin toss, are not realisable only assuming the authenticated channels model.

It becomes clear that the set-up assumptions have to be enhanced, if one should be able to construct UC secure protocols. Indeed, if the authenticated communication and a common reference string functionalities are given for free, then according to [CLOS02] any two party ideal functionality is realizable in the $\mathcal{F}_{CRS}^{\mathcal{D}}$-hybrid model. Other important results in the area state that two party functionalities such as $\mathcal{F}_{COM}^{C,R}$ or $\mathcal{F}_{ZK}^{R}$ can be UC realized also in the $\mathcal{F}_{CT}$-hybrid model, if authenticated communication is assumed [CF01, DN02].

If more efficient protocols are required, then having only set-up assumptions, even powerful ones, might not suffice. Strong cryptographic assumptions are also needed. A non-interactive protocol that realizes ideal commitment is described in [CF01]. Its security requires the above mentioned set-up assumptions plus the existence of trapdoor permutations. In this case, by non-interactive protocol we understand that both the commit and the opening phase consist of a single message sent from the committer to the recipient.

Finally, it is important to mention that, besides the UC framework described in this chapter, other models exist for defining and proving security of protocols by using the idea of simulatability. For example, the reactive simulatability notion, which at high level is similar to the UC notion, was first introduced in [PW00]. Further extensions to the inital framework were made by Backes,

Pfitzmann and Waidner [BPW04], where it was shown that reactive simulatability is preserved under universal composition.

Chapter 4

Long-term Universal Composability

In the previous chapter we have detailed the notion of universal composability. More precisely, we have seen that a secure protocol in this model maintains its security properties even when it interacts with other protocols, in large systems. A different way to look at the notion of security is by taking into consideration that no more information should be revealed to any adversary, possibly even unbounded, after the protocol is over. In the following, this will be called long-term security and it complements the universal composability property. We will discuss how long-term security and universal composability can be combined and we give an overview of the main results that were obtained in this area.

4.1 Long-term Security

As technology advances, attacks which are not possible yet, could be a threat at some point in the future for today's secure protocols. For example, consider a protocol which involves classified information, like medical data or government secrets. The security of such a protocol is nowadays based on assumptions that certain problems are computationally hard to solve by the existing technology. But an adversary gaining unlimited computational power in the future, may use the interaction he had with the protocol in order to reveal some party's secrets.

Thus, we want to ensure that secrets can not be uncovered, even after the protocol has ended and the adversary has become all powerful. This property, known as long term-security or everlasting security, has been considered before. One approach was to develop protocols that are secure in this model, if the adversary has a certain bound on the memory used. For example, a long-term secure key exchange protocol was constructed in the bounded storage model [CM97].

Other already existing cryptographic building blocks fulfill the long-term security property by definition. In a protocol that is zero-knowledge, only the verifier could, in theory, obtain more information after the interaction with the prover has ended. Thus, statistical zero-knowledge arguments [BCC88] are long-term secure: not even an unbounded verifier can break the statistical zero-knowledge property. In the same manner, perfectly hiding commitment schemes [NOVY98] are also long-term secure. The possibly corrupted sender can not change the committed value after the commitment has finished. Likewise, due to the strong hiding property, a corrupted verifier can not learn the actual commitment, before the unveil phase has been performed.

After giving a few examples where long-term security holds, it is worth mentioning that

the research conducted so far has hardly taken into consideration combining this property with a protocol composition operation. Indeed, the first study of long-term secure and universally composable cryptographic protocols has been done by Müller-Quade and Unruh. In [MQU07] they define the notion of long-term universal composability (or long-term UC) and describe a series of impossibility results as well as some protocols that fulfill the new notion. The next section contains a short overview of their work.

4.2 Long-term UC Security

The long-term UC security framework [MQU07] is built upon the already existing UC model. As explained in the previous chapter, there are two worlds. The real world contains the parties running a protocol π, its adversary and the environment machine $\mathcal{Z}$. In the ideal world, the ideal simulator interacts with an ideal functionality and with the environment. In order to give a definition that summarizes both the notion of long-term security and the universal composability property, it was taken into account that the adversary could become unlimited after the end of the protocol. Still, at this stage, he cannot send or receive any message to or from any party. Thus, intuitively, the machine which actually gains unconditional computational power after the protocol has finished should be the environment $\mathcal{Z}$.

At first glace, it seems as if two phases have to be modeled. During the first one, the protocol runs in the real world and the ideal $\mathcal{S}$ has to make the simulation in the ideal world. All the machines used so far are computationally bounded. In the second stage, no more messages are sent between the parties in the real world, thus only the environment, which has gained possibly unlimited computational power, has some role at this point in time. In [MQU07] it has been informally explained that actually the two above mentioned phases can equivalently be modeled as one phase. To obtain this, the environment is considered all powerful from the very beginning and we require that the views it gets from interacting with the machines in both worlds are statistically indistinguishable. This last property ensures that the difference between the distributions of the views does not depend any more on the power of the environment, but it reflects their statistical closeness.

Based on the above discussion, the following two definitions formally present the notion of long-term UC emulation for the case of general protocols and then, in particular, for an ideal process implementing an ideal functionality. As stated in chapter 3, by $EXEC_{\pi,\mathcal{A},\mathcal{Z}}\ (k,z)$ we denote the output of $\mathcal{Z}$ in an execution of the protocol π with adversary $\mathcal{A}$ and environment $\mathcal{Z}$, where k is the security parameter and z is the auxiliary input to the environment $\mathcal{Z}$. The family of random variables $\{EXEC_{\pi,\mathcal{A},\mathcal{Z}}\ (k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^{poly(k)}}$ is denoted by $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$. In an analogous way, for the case of the ideal process for a functionality $\mathcal{F}$ interacting with the simulator $\mathcal{S}$ and environment $\mathcal{Z}$, we have the notations $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}\ (k,z)$ and $EXEC_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ for the random variable representing the output of $\mathcal{Z}$ and for the family of these random variables, respectively.

Definition 23 (Long-term UC Emulation)**.** *Let π and ϕ be PPT protocols. Then π long-term UC emulates ϕ if for every PPT adversary $\mathcal{A}$ there is a PPT ideal simulator $\mathcal{S}$ such that for every environment $\mathcal{Z}$, possibly unbounded, the two families of random variables $EXEC_{\pi,\mathcal{A},\mathcal{Z}}$ and $EXEC_{\phi,\mathcal{S},\mathcal{Z}}$ are statistically indistinguishable.*

Definition 24 (Long-term UC Realizing a Functionality)**.** *Let $\mathcal{F}$ be an ideal functionality and let π be a protocol. We say that π long-term UC realizes $\mathcal{F}$ if π long-term UC emulates the ideal process for $\mathcal{F}$.*

The definition of the long-term UC security notion, as it is given above, differs very little from that of universal composability. This is why we expect that many results connected to UC security still hold in the long-term UC model. For example, the composition theorem remains valid. A sketch proof for it was given in [MQU07]. For the sake of completeness, we state the theorem here, together with its main consequence.

Theorem 25 (Long-term Universal Composition Theorem)**.** *Let π, ρ, ϕ be probabilistic polynomial time multi-party protocols such that ρ long-term UC emulates ϕ and π makes calls to protocol ϕ. Then protocol $\pi^{\rho/\phi}$ long-term UC emulates protocol π.*

Theorem 26 (Realizing Functionalities by Long-term Universal Composition)**.** *Let $\mathcal{F}$, $\mathcal{G}$ be ideal functionalities such that $\mathcal{F}$ is PPT. Let π be a protocol that long-term UC realizes $\mathcal{G}$ and makes calls to functionality $\mathcal{F}$. If ρ is a protocol that long-term UC realizes $\mathcal{F}$, then the composed protocol $\pi^{\rho/\mathcal{F}}$ long-term UC realizes functionality $\mathcal{G}$.*

4.3 Previous Results

In this section we review the main results obtained so far in the long-term UC model. We have seen that in order to construct secure universally composable protocols, we can not expect to succeed without assuming access to ideal functionalities such as, for example, the common reference string. The long-term UC notion is even more restrictive from this point of view. In [MQU07] it has been proved that it is impossible to build secure long-term UC commitments just by using any functionality contained in the class of only temporarily secret functionalities. This class gathers all ideal functionalities that have the following property: all secrets shared between a party and the ideal functionality can be inferred from the view of the other parties by an unbounded adversary. Still, all is not lost. In the same paper it was presented a protocol which given access to two instances of a $\mathcal{F}_{ZK}^{SAT}$ functionality, where SAT is the satisfiability relation, long-term UC realizes $\mathcal{F}_{COM}^{C,R}$.

Thus, the attention was turned towards realizing the $\mathcal{F}_{ZK}^{R}$ ideal functionality, for different relations R. Unfortunately, strong impossibility results were obtained also in this direction [MQU07]. Mainly, if the relation R is without an essentially unique witness, then no temporarily secret functionality can be used to implement a protocol realizing $\mathcal{F}_{ZK}^{R}$. However, this functionality was enough to build a protocol that realizes $\mathcal{F}_{ZK}^{Blum}$, where by *Blum* we understand the relation represented by the Blum integers and their factorization used as a witness.

A different approach was then taken by using a set-up assumption which can be replaced by an already existing device, the signature card (SC). This is a trusted hardware device that computes signatures on messages given by its owner, but never reveals the secret key. This property is based on a strong security requirement on the signature scheme used. Given the assumption that such a device is used [MQU07], protocols were constructed for implementing long-term UC commitments and zero-knowledge arguments for arbitrary relations. The proofs use the general assumption on the existence of one-way functions, which however does not allow for efficient constructions. Our goal in this work is to build on these ideas and construct an efficient long-term UC commitment scheme, even though this will imply using stronger cryptographic assumptions.

4.4 Further Remarks

Having in mind the set of impossibility results that were detailed above, one can wonder if the notion of long-term UC security was defined in a rather too restrictive way. Let the stand-alone version of the long-term UC definition be considered: the outputs of the parties and adversary, and

the outputs of the simulator should be statistically indistinguishable, but without involving any environment. As explained in [MQU07], if we want this minimal stand-alone security requirement together with the universal composability property, then the security notion obtained is similar to that presented in the long-term UC definition, with one difference: the simulator is allowed to depend on the environment. On the other hand, the fact that the simulator does not depend on the environment was never used in the proofs. Thus, the negative results would still hold even in an intuitively less restrictive setting.

The way the long-term UC notion was stated has another implication that we just mention here, without giving all the details. Based on a common reference string, in [DN02] a perfectly hiding commitment scheme, that is also secure in the UC model, has been constructed. Müller-Quade and Unruh show in [MQU07] that this commitment is actually not long-term UC secure. This may come as a surprising fact as it shows that long-term UC security is not completely equivalent to long-term security and universal composability considered together.

An important observation needs to be made. All the long-term UC secure protocols mentioned in the above sections were assuming secure channels. Also the adversarial model considered was static and active. These assumptions will be maintained also throughout the next chapter.

Chapter 5

An Efficient Long-term UC Secure Commitment

In the previous two chapters we have introduced the UC and the long-term UC models for defining and proving secure cryptographic properties. We have presented the most important known results about protocols that fulfill these properties. In particular, in [MQU07] it was shown that if one-way functions exist then long-term secure UC commitments can be designed, but without focusing on efficiency. In the following we will build upon their idea of using a signature card and add stronger cryptographic assumptions in order to construct a new long-term UC secure protocol that has plausible set-up assumptions and is efficient to use in practice.

5.1 Nontrivial Commitment Protocols

In the ideal model, the input received by an ideal party is sent immediately to the ideal functionality. Also, the output computed by the ideal functionality for each party respectively should be, intuitively, received in the same manner. On the other hand, the real adversary that attacks the protocol in the real world, can delay the transmission of messages between the parties. We need to be able to simulate this situation also in the ideal model. A natural way to tackle this task is to relax the properties of an ideal functionality as follows. Each time the ideal functionality wants to send a message v to one of the ideal parties, it first informs the ideal simulator about this action. If the message v is public, then the simulator also receives v. Finally, the ideal functionality sends v to the designated party only after the simulator has accepted the delivery of the message. In the following we assume this relaxation for any ideal functionality that is used, but for the sake of readability we do not explicitly mention it.

A direct consequence of the model above makes it possible that a protocol in which the parties do not communicate with each other and do not send any output to the environment $\mathcal{Z}$, securely implements any functionality. Indeed, the corresponding simulator for the ideal model just needs to deny permission for any message that should be sent by the ideal functionality to the ideal parties. Obviously, this real-world protocol is not interesting for any practical purpose.

Our goal is to construct an efficient long-term secure UC protocol that emulates the ideal commitment functionality described in chapter 3. Taking into consideration the observations made above, we want this protocol to have an additional property. Namely, we need to construct a protocol that in case neither the sender, nor the recipient is corrupted and all the messages are delivered, then the recipient will accept both the commit and the unveil phase of the protocol.

Such a protocol is called *nontrivial protocol*.

Definition 27 (Nontrivial Protocol)**.** *A commitment protocol π in which the recipient accepts both the commit and the unveil phase with overwhelming probability, if no party is corrupted and all messages are delivered by the real-model adversary, is called a nontrivial commitment protocol.*

5.2 Possible Ideal Functionalities

As discussed in the previous chapters, in order to build a UC protocol, and even more so, a long-term UC one, we cannot expect to do it without the use of an ideal functionality. Still, there are problems that arise when choosing among different possible set-up assumptions. On the one hand, we have to design and use functionalities for which real-life replacements exist, such as protocols or devices that can successfully substitute them in order to obtain a protocol usable in a real-life system. On the other hand, the chosen functionalities should make the protocol as efficient as possible. More precisely, it is desired to restrict the number of instances of the functionalities used as well as the number of calls we make to any such functionality. Moreover, the complexity of the overall protocol, measured by the number of messages sent between parties, the running time and the total length of communication should also be taken into consideration.

Building upon the technique elaborated by Müller-Quade and Unruh in [MQU07], the long-term UC protocol that we will construct makes use of an ideal functionality called signature card. First, we take a look at the signature card device which already exists on the market. Intuitively, a signature card is a trusted device which has precisely one owner. Upon request, the owner can ask the signature card to digitally sign any message. For this purpose, the signature card has a built-in pair of secret key and public key, which are used according to a signature scheme implemented within the signature card. Moreover, the public key can be obtained from a trusted authority, while the secret key is never revealed, not even to the owner of the card. This intuition is formalized next, as presented in [MQU07].

Definition 28 (Ideal Functionality Signature Card)**.** *Let $\Sigma = (Gen, Sig, Verify)$ be a signature scheme. Let H be a party. Then the functionality $\mathcal{F}_{SC}^{H,\Sigma}$ representing a signature card for scheme Σ with holder H, behaves as follows:*

- *Upon the first activation, $\mathcal{F}_{SC}^{H,\Sigma}$ chooses a public and secret key pair (pk, sk) using the key generation algorithm $Gen(1^k)$, where k is the security parameter for the signature scheme Σ.*
- *Upon a message (public key) from a party P or the adversary, $\mathcal{F}_{SC}^{H,\Sigma}$ sends pk to that party or adversary, respectively.*
- *Upon a message $(sign, m)$ from the holder H, $\mathcal{F}_{SC}^{H,\Sigma}$ produces a signature σ for m using the secret key sk and sends σ to H.*

It is important to note that an ideal functionality $\mathcal{F}_{SC}^{H,\Sigma}$ can be directly accessed only by the parties involved in the protocol and by the possible adversaries that control them. This is why the environment has only indirect access to such a functionality and this implies, for example, that it can ask for having a message signed only through an adversary that has corrupted the owner of a signature card used in the protocol.

Having in mind the goal of obtaining an efficient commitment protocol in the long-term UC model, it is clear that the overall efficiency of a protocol designed in the $\mathcal{F}_{SC}^{H,\Sigma}$-model depends on the choice of the signature scheme as well. Hence, the signature scheme should have a very good efficiency if we take into account the length of the signatures obtained. In order to maintain

the long-term UC property of the protocol we build, a strong security guarantee, like EF-CMA security, needs to be fulfilled by the signature scheme. Moreover, the signatures obtained should allow the owner of the signature card to use efficient techniques for proofs of knowledge.

A suitable scheme that achieves the properties mentioned above is the Camenish-Lysyanskaya digital signature scheme [CL02]. An important point about this scheme is that it allows the owner to obtain signatures not only on individual messages, but also on blocks of messages. The use for this extra property becomes clear when following the actual security proof of the long-term UC protocol we will bulid. As during the protocol only pairs of messages need to be signed, the following description of the scheme takes this fact into account. A modification that we make to the original digital signature scheme from [CL02] is that we fix the number of possible incoming messages to utmost two per signing round.

It is also important to mention that in the original scheme the different parameters used had no dependency between them, thus, they can be chosen to verify certain relations without affecting the security of the overall construction. We use this property, as it will be seen below.

The Camenish-Lysyanskaya scheme for signing pairs of messages will be denoted by CL_2 and consists of three algorithms Gen, Sig and Verify representing the key generation, signing and verification algorithm, respectively. They work as follows.

- Key generation: on input 1^k it chooses $n = p \cdot q$, with $p = 2 \cdot p' + 1$ and $q = 2 \cdot q' + 1$. The values p, q, p' and q' are all primes and p and q have length k. It then takes $a_1, a_2, b, c \xleftarrow{r} \mathrm{QR}_n$, set $sk = p$ and output $pk = (a_1, a_2, b, c)$. Let l_n denote the length of the modulus n, where $l_n \geq 2 \cdot k - 1$.

- Message space: let l_m be a parameter such that $l_m \leq l_n$. The message space consists of all binary strings of length l_m for which the following holds: $2^{l_m - 1} - 2^{l_m - 2 \cdot k - 3} \leq m \leq 2^{l_m - 1} + 2^{l_m - 2 \cdot k - 3}$.

- Signing algorithm: on input (m_1, m_2), it chooses a random prime $e \geq 2^{l_m + 2}$ of length $l_e \geq l_m + 2$ such that $2^{l_e - 1} + 2^{l_e - 2} - 2^{l_e - 2 \cdot k - 4} \leq e \leq 2^{l_e - 1} + 2^{l_e - 2} + 2^{l_e - 2 \cdot k - 4}$. It also chooses a random number s of length $l_s = l_n + l_m + k$, where k is the security parameter. Finally, using the fact that it knows the secret key p, Sig computes v such that

  $$v^e = {a_1}^{m_1} \cdot {a_2}^{m_2} \cdot b^s \cdot c \bmod n$$

  and sets $\sigma = (v, e, s)$.

- Verification algorithm: to verify that the tuple $\sigma = (v, e, s)$ is a signature on the pair of messages (m_1, m_2), Verify checks that $v^e = {a_1}^{m_1} \cdot {a_2}^{m_2} \cdot b^s \cdot c \bmod n$ and check that $e > 2^{l_e - 1}$.

From the security point of view, in [CL02] it was proved that the CL_2 signature scheme is existentially unforgeable against chosen message attacks if the strong RSA assumption holds with respect to special RSA moduli. We further note that in the original scheme the values m and e could be any values of length l_m and l_e, with e being additionally a prime number. The restriction we have taken in the description above does not lower the security of the signature scheme if we consider l_m and l_e to be reasonably larger than k. For example one can assume $l_m \geq 4k$ and also $l_e \geq 4k$.

The second primitive, used for constructing our long-term UC protocol, is a commitment scheme elaborated by Damgård and Fujisaki in [DF02]. There it is shown that the commitment scheme is statistically hiding and computationally binding if the factoring assumption holds. Their construction is presented next.

- Key generation: the public key pk consists of a special RSA modulus n of length l_n, $a_2 \xleftarrow{r} QR_n$ and $a_1 \xleftarrow{r} \langle a_2 \rangle$, where $\langle a_2 \rangle$ represents the group generated by a_2.

- Commitment: the commitment for a value m of length l_m with randomness $r \xleftarrow{r} \mathbb{Z}_n$, is the value ${a_1}^m \cdot {a_2}^r \bmod n$.

It is important to note that for the hiding property, the only requirement for the public key is that $a_2 \in \langle a_1 \rangle$. This condition will be fulfilled by the protocol we use. Indeed, we are going to derive the commitment public key from the public key generated by a signature card functionality implementing a slight variation of the CL_2 scheme where the relation $a_1 \in \langle a_2 \rangle$ holds.

Another clarification concerns the fact that if the strong RSA assumption holds, then factoring is hard. A sketch proof for this fact was given in chapter 2. For the security of the two above building blocks this means it is sufficient only that the strong RSA assumption holds.

Finally, for our construction we use a perfectly binding and computationally hiding commitment scheme. In the proofs we make, an explicit form of the commitment is not needed, but only the property that the scheme used is perfectly binding. Thus, any commitment scheme with this property can be used here. For example, one can take the efficient construction described in [CD98]. The security of this scheme relies on an slightly stronger assumption than the existence of one-way functions.

5.3 An Efficient Long-term Secure UC Commitment

Our goal is to construct a long-term UC commitment in the $\mathcal{F}_{CL_2}^{C,SC}$-hybrid model. The sender in this commitment will be the owner of the signature card $\mathcal{F}_{CL_2}^{C,SC}$ and during the protocol it will make only one call to it for signing a pair of messages, in the commit phase. In the unveil phase, the sender will only need to reveal the recipient the pair it has signed.

As a first step towards this construction, we will use a slightly modified version of the Camenish-Lysyanskaya signature scheme CL_2. The difference is that the key generation algorithm Gen will not choose a_1 randomly from QR_n. But, after choosing a_2 in the usual way, mainly by $a_2 \xleftarrow{r} QR_n$, the modified generation algorithm will compute a value $d \xleftarrow{r} \mathbb{Z}_{\phi(n)}$ and finally, let $a_1 = {a_2}^d \bmod n$. The rest of the key generation, signing and verifying algorithms remains as in CL_2. We denote this new signature scheme by CL_2^*. Our next lemma shows that the pair of keys (pk, sk) generated by CL_2 are statistically indistinguishable from those generated by CL_2^*.

Lemma 29 (Statistical Indistinguishability between CL_2 and CL_2^*)**.** *Let $T_1 = \{(pk, sk) : (pk, sk) \leftarrow \mathrm{CL}_2\}$ and $T_2 = \{(pk, sk) : (pk, sk) \leftarrow \mathrm{CL}_2^*\}$ be the two distributions of the pairs of keys (pk, sk) obtained by using CL_2 and CL_2^*, respectively. Then T_1 and T_2 are statistically indistinguishable.*

Proof. The one difference between the two signature schemes is in the way value a_1 is chosen as part of the public key. In the case of CL_2^*, a_1 depends only on value a_2, with the rest of the values being independently chosen from a_1 and a_2. Thus, we need to show only that the distributions of pairs (a_1, a_2) taken accordingly to the key generation algorithms of the two schemes CL_2 and CL_2^* are statistically indistinguishable. Intuitively, we have to prove that the random distribution on QR_n is statistically indistinguishable from that defined by choosing $a_1 = {a_2}^d$, where d is random in $\mathbb{Z}_{\phi(n)}$ and a_2 follows the random distribution also on QR_n.

More formally, let $D_1 = \{(a_1, a_2) : a_1, a_2 \xleftarrow{r} QR_n\}$ and let $D_2 = \{(a_1, a_2) : a_2 \xleftarrow{r} QR_n, d \xleftarrow{r} \mathbb{Z}_{\phi(n)}, a_1 = {a_2}^d\}$. According to the exposition made above, it is sufficient to show that D_1 and D_2 are statistically indistinguishable. As described in the lemma 14, from chapter 2, if n is a

special RSA modulus, like the one generated in both CL_2 and CL_2^*, then QR_n is a cyclic group under multiplication, of size $p' \cdot q'$, where all but $p' + q'$ of the elements are generators. In terms of probabilities this means that:

$$Pr(\langle a_2 \rangle = QR_n | a_2 \xleftarrow{r} QR_n) = \frac{p' \cdot q' - p' - q'}{p' \cdot q'} = 1 - \frac{1}{p'} - \frac{1}{q'}$$

The probability computed above equals also the probability that a_1 is uniformly distributed in QR_n when a_1 is chosen by the key generation algorithm of CL_2^*. More formally:

$$Pr(a_1 \text{ uniformly distributed in } QR_n | a_2 \xleftarrow{r} QR_n, d \xleftarrow{r} \mathbb{Z}_{\phi(n)}, a_1 = {a_2}^d) = Pr(\langle a_2 \rangle = QR_n | a_2 \xleftarrow{r} QR_n)$$

Let D be any algorithm, possibly computationally unbounded, and let $|n| = 2 \cdot k$ as it is also specified in both key generation algorithms for CL_2 and CL_2^*. Since p and q are both primes of length k and $p = 2 \cdot p'$ and $q = 2 \cdot q'$ it implies that $p, q \geq 2^{k-3}$. With loss of generality, assume that $p \leq q$. If D tries to distinguish between the distributions D_1 and D_2, he can succeed, according to the relations above with at most the following probability:

$$|Pr(D(x) = 1 | x \leftarrow D_1) - Pr(D(x) = 1 | x \leftarrow D_2)| \leq \frac{1}{p'} + \frac{1}{q'} \leq \frac{2}{p'} \leq \frac{1}{2^{k-2}}$$

The last value represents a negligible function in the security parameter k, which means D_1 and D_2 are statistically indistinguishable. This, in turn, implies that the keys generated by CL_2 and CL_2^* are also statistically indistinguishable. □

We are now ready to state the main result of this work.

Theorem 30 (Efficient Long-term Secure UC Commitment using $\mathcal{F}_{SC}^{C,CL_2^*}$)**.** *Assume that the strong RSA assumption holds. Then there is a nontrivial and efficient protocol π that long-term UC realizes $\mathcal{F}_{COM}^{C,R}$ and uses one instance of an ideal functionality $\mathcal{F}_{SC}^{C,CL_2^*}$.*

The protocol π we propose looks as follows:

- In the commit phase, the sender C is activated by the environment $\mathcal{Z}$ with input α. Then C uses the public key $pk = (n, a_1, a_2, b, c)$ generated by the ideal functionality $\mathcal{F}_{SC}^{C,CL_2^*}$ in order to obtain a public key (n, a_1, a_2) for the Fujisaki-Damgård commitment scheme. By choosing $\beta \leftarrow \mathbb{Z}_n$, C computes the commitment $COM = a_1^{\alpha} \cdot a_2^{\beta}$ and sends COM to the recipient R.

- The sender C obtains a signature $\sigma = (e, s, v)$ from $\mathcal{F}_{SC}^{C,CL_2^*}$ on the pair (α, β). Then C chooses at random values ω and r_ω of length l_n and computes

  $$C_v = v \cdot {a_1}^{\omega} \bmod n$$

  and

  $$C_\omega = {a_1}^{\omega} \cdot {a_2}^{r_\omega} \bmod n.$$

  Next, C sends C_v and C_ω to R.

- The sender C performs the following proof of knowledge (the actual details on how this is done are explicitly stated after the protocol description):

$$
\begin{aligned}
&PK(\alpha, \beta, \varepsilon, \sigma, \pi, \varphi, \gamma, \delta) : \\
&\quad COM \cdot c = C_v^{\varepsilon} \cdot \left(\frac{1}{b}\right)^{\sigma} \cdot \left(\frac{1}{a_1}\right)^{\pi} \wedge C_\omega = a_1^{\gamma} \cdot a_2^{\delta} \wedge \\
&\quad 1 = C_\omega^{\varepsilon} \cdot \left(\frac{1}{a_1}\right)^{\pi} \cdot \left(\frac{1}{a_2}\right)^{\varphi} \wedge COM = a_1^{\alpha} \cdot a_2^{\beta} \wedge \\
&\quad 0 \leq \alpha < 2^{l_m} \wedge \\
&\quad 0 \leq \beta < 2^{l_n} \wedge \\
&\quad 2^{l_e - 1} < \varepsilon \leq 2^{l_e}
\end{aligned}
$$

- If the recipient R accepts the proof, then he sends (accept) to the environment $\mathcal{Z}$, otherwise R aborts.
- In the unveil phase, i.e. upon input *unveil* from $\mathcal{Z}$, the sender C sends α and β to the recipient.
- The recipient R checks whether $COM = a_1^{\alpha} \cdot a_2^{\beta}$ and if this holds, it sends $(accept, \alpha)$ to the environment $\mathcal{Z}$. Otherwise, R does nothing.

The proof of knowledge $PK(\alpha, \beta, \varepsilon, \sigma, \pi, \varphi, \gamma, \delta)$, developed in [CL02] without including the use of perfectly binding commitment scheme, has the description detailed below:

- In the first step, the recipient chooses the values e_1', e_2', $e_3' \xleftarrow{r} [0, 2^k)$ and using the perfectly binding commitment scheme, R commits to each of them. It thus obtains c_1, c_2 and c_3 which it sends to C.
- The second step involves the following actions from the sender C:

  It takes the random values:

  - $r_1 \xleftarrow{r} [0, 2^{l_e} - 1]$
  - $r_2 \xleftarrow{r} [0, 2^{l_s} \cdot 2^{2k} - 1]$
  - r_3, $r_6 \xleftarrow{r} [0, 2^{l_e} \cdot 2^{l_n} \cdot 2^{2k} - 1]$
  - r_4, $r_5 \xleftarrow{r} [0, 2^{l_n} \cdot 2^{2k} - 1]$
  - $r_7 \xleftarrow{r} [0, 2^{l_m} - 1]$
  - $r_8 \xleftarrow{r} [0, 2^{l_n} - 1]$
  - z_1, z_2, $z_3 \xleftarrow{r} [0, 2^k)$

  Then C computes:

  - $d_1 = C_v^{r_1} \cdot \left(\frac{1}{b}\right)^{r_2} \cdot \left(\frac{1}{a_1}\right)^{r_3} \bmod n$
  - $d_2 = a_1^{r_4} \cdot a_2^{r_5} \bmod n$

- $d_3 = C_\omega^{r_1} \cdot \left(\frac{1}{a_1}\right)^{r_3} \cdot \left(\frac{1}{a_2}\right)^{r_6} \bmod n$
- $d_4 = a_1^{r_7} \cdot a_2^{r_8} \bmod n$

and sends d_1, d_2, d_3, d_4 and z_1, z_2, z_3 to the recipient R.

- In the third step, the recipient unveils the commitments c_1, c_2, c_3 to the values e'_1, e'_2, e'_3, respectively, which are then sendt to C.
- In the fourth step, the sender C computes the challenges $e_1 = e'_1 + z_1 \bmod 2^k$, $e_2 = e'_2 + z_2 \bmod 2^k$ and $e_3 = e'_3 + z_3 \bmod 2^k$ which C uses to calculate the following values in $\mathbb{Z}$. These values are then sent to the recipient R:
  - $s_1 = r_1 + e_1 \cdot e$
  - $s_2 = r_2 + e_1 \cdot s$
  - $s_3 = r_3 + e_1 \cdot e \cdot \omega$
  - $s_4 = r_4 + e_2 \cdot \omega$
  - $s_5 = r_5 + e_2 \cdot r_\omega$
  - $s_6 = r_6 + e_1 \cdot e \cdot r_\omega$
  - $s_7 = r_7 + e_3 \cdot \alpha$
  - $s_8 = r_8 + e_3 \cdot \beta$

  After receiving these values, R checks if the relations listed below are satisfied. If all of them are true, then he accepts the proof, otherwise he rejects.
  - $C_v^{s_1} \cdot \left(\frac{1}{b}\right)^{s_2} \cdot \left(\frac{1}{a_1}\right)^{s_3} = (COM \cdot c)^{e_1} \cdot d_1 \bmod n$
  - $a_1^{s_4} \cdot a_2^{s_5} = C_\omega^{e_2} \cdot d_2 \bmod n$
  - $C_w^{s_1} \cdot \left(\frac{1}{a_1}\right)^{s_3} \cdot \left(\frac{1}{a_2}\right)^{s_6} = d_3 \bmod n$
  - $a_1^{s_7} \cdot a_2^{s_8} = COM^{e_3} \cdot d_4 \bmod n$
  - $s_1 \in [0, 2^{l_e} - 1]$
  - $s_7 \in [0, 2^{l_m} - 1]$
  - $s_8 \in [0, 2^{l_n} - 1]$
  - $s_2 \in [0, 2^{l_s} \cdot 2^{2 \cdot k} + 2^{l_s} \cdot 2^k - 2^k - 1]$
  - $s_3,\ s_6 \in [0, 2^{l_e} \cdot 2^{l_n} \cdot 2^{2 \cdot k} + 2^{l_e} \cdot 2^{l_n} \cdot 2^k - 2^k - 1]$
  - $s_4,\ s_5 \in [0, 2^{l_n} \cdot 2^{2 \cdot k} + 2^{l_n} \cdot 2^k - 2^k - 1]$

After describing in detail the protocol π and the proof of knowledge PK we proceed to show that the statement from the theorem 30 indeed holds.

Proof. First of all, we prove that the protocol π is nontrivial. This holds since if none of the parties is corrupted and all the messages are delivered by the real-model adversary, than the recipient will accept the commitment with overwhelming probability.

Note that considering only one party to be corrupted and not both in the same time is sufficient. In the event that both the sender and the recipient are controlled by the real-model adversary, this is irrelevant when static corruption is considered.

Next, we need to show that for each of the two parties, sender and recipient being corrupted, an appropriate simulator $\mathcal{S}$ can be constructed in the ideal world in order to fulfill the long-term UC definition. We start by studying the case of the corrupted sender in the real model. This is summarized in the following lemma:

Lemma 31 (Statistical Indistinguishability in case of Corrupted Sender)**.** *Let π be the protocol described above, and let the assumptions be those stated in theorem 30. If the sender is corrupted, then for every adversary $\mathcal{A}$ controlling C there exist a simulator $\mathcal{S}$ such that the views of the environment $\mathcal{Z}$ obtained from the real and ideal world, respectively, are statistically indistinguishable, for every $\mathcal{Z}$, possibly unbounded.*

Proof. The ideal simulator $\mathcal{S}$ is constructed in the following way:

- $\mathcal{S}$ runs a copy of the real adversary $\mathcal{A}$ and he connects it to :
  - the environment $\mathcal{Z}$
  - a simulated copy of the real recipient R
  - a simulated instance of the functionality $\mathcal{F}_{SC}^{C,CL_2^*}$ with the only modification that all the pairs received to be signed are also stored.
- If the simulated recipient R accepts the commit phase, then $\mathcal{S}$ extracts a witness α for the commitment by checking whether there is a pair (α, β) among the pairs stored by $\mathcal{F}_{SC}^{C,CL_2^*}$ such that $COM = a_1^\alpha \cdot a_2^\beta$. If such a pair (α, β) exists, then $\mathcal{S}$ uses α in the following. Otherwise $\mathcal{S}$ aborts.
- $\mathcal{S}$ sends $(commit, \alpha)$ to the simulated $\mathcal{F}_{COM}^{C,R}$ as a corruption message such that the ideal functionality uses it as coming from the dummy party C.
- If the simulated recipient R accepts the unveil phase, then $\mathcal{S}$ sends $(unveil)$ to the ideal functionality $\mathcal{F}_{COM}^{C,R}$.

We show that the view of the environment $\mathcal{Z}$ in the case it is interacting in the real world with the corrupted sender and the honest recipient is statistically indistinguishable from the view of the environment interacting in the ideal world with the ideal simulator, described above, and the ideal functionality $\mathcal{F}_{COM}^{C,R}$. Note that $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$, hence the direct communication between the environment $\mathcal{Z}$ and the adversary in the real world and that between $\mathcal{Z}$ and the simulated adversary run by $\mathcal{S}$ in the ideal world are identically distributed. Moreover, in the real-world model, the view of the environment consists also of the messages sent from the honest recipient R to $\mathcal{A}$ and the output messages that R makes to $\mathcal{Z}$. In the ideal world, the environment receives the messages sent from the simulated R to the simulated $\mathcal{A}$ plus the messages that the honest dummy party R outputs, through the functionality $\mathcal{F}_{COM}^{C,R}$.

Thus, it is sufficient to prove that the environment's view of the output messages of party R, in the real world, and the view of the output messages of the ideal functionality, in the ideal world, are statistically indistinguishable. Intuitively, in order to do this we have to prove that the following two conditions are not fulfilled only with negligible probability.

The first condition is that, if the simulated R accepts the commit phase, than $\mathcal{S}$ needs to extract a witness α from the calls $\mathcal{A}$ has placed to the simulated functionality $\mathcal{F}_{SC}^{C,CL_2^*}$ and send it to the $\mathcal{F}_{COM}^{C,R}$ functionality, which will trigger the message $(accept)$ to be sent to $\mathcal{Z}$. Taking into account the construction of $\mathcal{S}$, this is equivalent to the following condition: if the simulated R accepts the commit phase, then $\mathcal{S}$ does not abort.

The second condition is that if the simulated recipient R accepts the unveil message (α', β'), then the value α' should equal the value α received by the ideal functionality $\mathcal{F}_{COM}^{C,R}$ from $\mathcal{S}$ at the end of the commit phase.

The reason why the two views of the environment $\mathcal{Z}$ are statistically indistinguishable, if both the above conditions are fulfilled by $\mathcal{S}$ with overwhelming probability, is the following. Since $\mathcal{S}$ does not abort, it means it was able to extract a witness (α, β) for the commitment COM. The second condition ensures that the same pair (α, β) was send by $\mathcal{S}$ to the simulated R when it has accepted the unveil phase. Thus, the value α sent by the ideal functionality to the environment in the unveil phase is, with overwhelming probability, the same as the value received by $\mathcal{Z}$ from the interaction with the parties of π and adversary $\mathcal{A}$.

We turn our attention now towards proving the first condition. For the clarity of the following proof, we first give some simplifying notation. Intuitively, we split the honest recipient R into three ITMs as described bellow. The first ITM R_1 receives as input a commitment value COM and then it halts without any output. The internal state of R_1 is set to the value COM. The second ITM R_2 plays the role of the verifier in the proof PK. In the end, R_2 either outputs $(accept)$ or it aborts. The internal state of R_2 is set also to COM. Finally, the ITM R_3 receives an unveil message (α, β) and checks whether $COM = a_1^{\alpha} \cdot a_2^{\beta}$ holds and in this case it outputs $(accept, \alpha)$. Otherwise it ends without any output. The pair (α, β) is added to the internal state of R_3. It is clear that if we run R_1, R_2 and R_3 sequentially we obtain an exact simulation of R.

After clarifying the notation, we proceed with the actual proof. Our first condition translates to the following statement: if R_2 accepts the proof PK, then $\mathcal{S}$ aborts only with negligible probability. Let M be an ITM that groups together all the ITMs in the ideal world, besides the simulated R_2. Thus M contains the environment $\mathcal{Z}$, the simulated adversary $\mathcal{A}$, the simulated ITMs R_1 and R_3 and the ideal simulator $\mathcal{S}$. When M receives an input containing a value α, it lets $\mathcal{A}$ compute a value COM and it outputs it. Then M interacts with R_2 for what this ITM expect to be the proof PK. Thus the interaction between M and R_2 represents a perfect simulation of the ideal protocol that takes place in the ideal world. It it important to note that M has only oracle access to the signature scheme CL_2^*.

We denote by $\langle M, R_2 \rangle = 1$ the event that R_2 accepts the proof PK when interacting with M. It is clear that the probability for $\mathcal{S}$ to abort equals now the probability that $\langle M, R_2 \rangle = 1$ and M did not find any pair of messages (α_i, β_i) among the signing queries that were made to the signature card $\mathcal{F}_{SC}^{C,CL_2^*}$.

The following modifications to M keep this probability unchanged. First of all, the probability does not depend on any output made by the simulated R_3 which is contained in M. Thus, this output is simply not made by the modified M. Secondly, since M cannot find a witness pair for the commitment COM among the queries to $\mathcal{F}_{SC}^{C,CL_2^*}$, we can assume the modified M does not ask $\mathcal{F}_{SC}^{C,CL_2^*}$ to sign any pairs that represent a witness for the commitment COM. The ITM obtained from M with the above modifications will be denoted by M'. In terms of probabilities we have: the probability that $\mathcal{S}$ aborts equals the probability that $\langle M', R_2 \rangle = 1$.

Assume by contradiction that $\mathcal{S}$ aborts with non-negligible probability. Then, the probability that $\langle M', R_2 \rangle = 1$ is also not negligible. In [CL02] it was shown that a very similar proof to our PK is actually a proof of knowledge. The only difference we introduce here is that the recipient R_2 first commits to its challenges and later it unveils them. However, their proof applies here also since for the proof-of-knowledge property the verifier is considered honest. Indeed, in our proof PK, the challenges committed to by the honest recipient are random values. This is also the case in [CL02].

Thus, by the proof-of-knowledge property, and using the fact that $\langle M', R_2 \rangle = 1$ with non-negligible probability, it means that there exist a knowledge extractor K which, given oracle access

to M', outputs a witness for the proof PK with non-negligible probability. Since M' has oracle access to the signature scheme CL_2^*, by regrouping the components inside K we can construct an ITM $\bar{K}$ which is equivalent to K, only that it has also oracle access to CL_2^*. Since $\bar{K}$ is a knowledge extractor for PK it outputs with not negligible probability a witness (α, β, s, e, v) such that the following hold: $COM = a_1^\alpha \cdot a_2^\beta \bmod n$ and $v^e = a_1^\alpha \cdot a_2^\beta \cdot b^s \cdot c \bmod n$ and $e \in [2^{l_e-1}, 2^{l_e})$. In other words, this means that $\bar{K}$ outputs a witness (α, β) for the commitment COM and for this witness he finds a valid signature (v, e, s). We call this property $(*)$.

On the other hand, by definition of M', this ITM cannot send to CL_2^* as a signing query any pair (α, β) that represents an unveil information for COM. The same property holds for $\bar{K}$, since it can query CL_2^* only through M'. Combining this with the property $(*)$ we obtain that $\bar{K}$, given only oracle access to CL_2^*, outputs a signature on a witness pair (α, β) for the commitment COM. In the same time, a signature for (α, β) could not have been queried to $\mathcal{F}_{SC}^{C,CL_2^*}$. This implies that $\bar{K}$ breaks the EF-CMA security of the signature scheme CL_2^* which is a contradiction to the EF-CMA security of CL_2 and lemma 29. Thus, we have shown that the first condition stated in the beginning of the proof, mainly that $\mathcal{S}$ aborts if R accepts the commitment COM, holds with negligible probability.

Next, we show that the second condition holds. We know that the recipient R_3 has accepted the unveil message (α', β'). This implies that first of all $\mathcal{S}$ did not abort and also $COM = a_1^{\alpha'} \cdot a_2^{\beta'}$ holds. Since $\mathcal{S}$ did not abort it means it had extracted a witness (α, β) from the signature card functionality such that $COM = a_1^\alpha \cdot a_2^\beta$ holds. If we assume by contradiction that $\alpha \neq \alpha'$ with non-negligible probability, it means that given a commitment COM the simulator $\mathcal{S}$ can be used as an adversary to break with non-negligible probability the binding property of the Fujisaki-Damgård commitment scheme. Since this commitment scheme is computationally binding and the trapdoor d is not used, we have obtained a contradiction. Thus we prove that the values α and α' are different only with negligible probability. This ends the proof of our second condition and also the proof for this lemma. □

Next, we investigate the case when the real-model recipient is corrupted. Similarly to the case above, we have state a lemma that summarize it:

Lemma 32 (Statistical Indistinguishability in case of Corrupted Recipient)**.** *Let π be the protocol described for theorem 30. If the recipient is corrupted, then for every adversary A controlling R there exist a simulator $\mathcal{S}$ such the views of the environment $\mathcal{Z}$ obtained from the real and ideal world are statistically indistinguishable, for every $\mathcal{Z}$, possibly unbounded.*

Proof. In this case, the ideal simulator $\mathcal{S}$ performs the following actions:

- $\mathcal{S}$ runs a simulated copy of the real adversary $\mathcal{A}$, which it connects to
  - the environment $\mathcal{Z}$
  - a simulated copy C of an honest sender, with some modifications, as described below
  - a simulated instance of the functionality $\mathcal{F}_{SC}^{C,CL_2^*}$.
- If $\mathcal{S}$ receives (*commit*) from the ideal functionality $\mathcal{F}_{COM}^{C,R}$, then it lets the simulated C construct a commitment COM' to 0.
- If $\mathcal{S}$ receives $(unveil, \alpha)$ from $\mathcal{F}_{COM}^{C,R}$, it computes β' such that $COM' = a_1^\alpha \cdot a_2^{\beta'}$. In order to do this, $\mathcal{S}$ uses the fact that it knows d such that $a_1 = a_2^d$. Then, $\mathcal{S}$ does not allow the simulated C to send the value 0 and the unveil information for COM' to $\mathcal{A}$. Instead, $\mathcal{S}$ sends the pair (α, β') to $\mathcal{A}$ as coming from C.

We have to prove that the view of the environment $\mathcal{Z}$ in the case it is interacting in the real world with the honest sender and the corrupted recipient is statistically indistinguishable from the view of the environment interacting in the ideal world with the ideal simulator, which was detailed above, and functionality $\mathcal{F}_{COM}^{C,R}$.

For the readability and clarity of the proof we intuitively "split" the honest sender C into three machines that run sequentially will give the exact simulation of an honest sender C. We denote these machines as described next. Let C_1 be an ITM such that on an input α, will use the statistically hiding commitment scheme to compute a commitment COM to α and then send COM to the recipient. The internal state of C_1 is set to (α, β, COM), where β is the unveil value for COM.

In a similar way, C_2 denotes an ITM with the following properties. Given the messages α, β and COM as internal state, the ITM C_2 asks the ideal functionality $\mathcal{F}_{SC}^{C,CL_2^*}$ to sign the pair (α, β). Using this signature, C_2 performs the proof PK as it was detailed in the protocol π. The internal state remains unchanged.

Finally, C_3 is an ITM such that on input $(unveil, \alpha)$ and as internal state (α, β, COM), sends (α, β) to the recipient. It is clear that if we run C_1, C_2 and C_3 sequentially, then this is an exact simulation of the honest sender C described in the protocol π.

Having the above notation in place, we can proceed with the actual proof. First we note that since $\mathcal{S}$ simulates a copy of the real adversary, the direct communication between $\mathcal{Z}$ and the simulated adversary, which is run by $\mathcal{S}$, in the ideal world, and the communication between $\mathcal{Z}$ and $\mathcal{A}$ in the real world, are identically distributed. Hence, it is sufficient if we consider that the view of the environment in the real world consists only of communication messages sent from the honest sender C to the adversary $\mathcal{A}$ and the output messages that $\mathcal{Z}$ receives directly from C.

We start by looking at the view of the environment $\mathcal{Z}$ when interacting with C_1 and $\mathcal{A}$ in the real world. Moreover, the environment does not receive any output messages from C_1 in the real world, but through $\mathcal{A}$ it receives all the communication between C_1 and $\mathcal{A}$. In case $\mathcal{Z}$ sends an input α to C_1, this communication is represented by the commitment value COM to the value α. On the other hand, in the ideal world, the communication between the simulated copy of C_1 and the simulated copy of adversary $\mathcal{A}$ is represented by a commitment to the value 0. Nevertheless, the commitment scheme used is statistically hiding, thus not even an unlimited environment $\mathcal{Z}$ can distinguish with more than negligible probability between the two commitments. Hence, so far, the views of the environment in the real and in the ideal world are statistically indistinguishable.

Next, we have to consider the interaction of $\mathcal{Z}$ with C_2 and $\mathcal{A}$ as it is done in the real world. This represents the proof $PK(\alpha, \beta, \varepsilon, \sigma, \pi, \varphi, \gamma, \delta)$ together with the messages C_v and C_ω that are sent from C_2 to $\mathcal{A}$. The witness for this proof is given by the set of values $\alpha, \beta, \varepsilon, \sigma, \pi, \varphi, \gamma, \delta$. In the following we will show that PK is actually a proof of knowledge and the distribution of C_v and C_ω in the real world (and similarly of the commitments C'_v and C'_ω computed by the simulated C_2 in the ideal world) are statistically close to the random distribution on $QR_n \times QR_n$.

Firstly, all the bases used to make the commitments in the first step of the proof PK are invertible elements in $\mathbb{Z}_n$. Actually, a more general statement holds: a random value $v \in \mathbb{Z}_n$ is invertible mod n with overwhelming probability. Indeed, the number of invertible elements in $\mathbb{Z}_n$, where $n = p \cdot q$ is an RSA modulus, is $\phi(n) = (p-1) \cdot (q-1)$. As proved before, in lemma 29, the difference $1 - \dfrac{\phi(n)}{n}$ is a negligible function in the length of n. This implies that only a negligible fraction of all element in $\mathbb{Z}_n$ are not invertible mod n. For a special RSA modulus n we have that the number of elements in QR_n is $\dfrac{\phi(n)}{4}$. Thus, a random value $v \in QR_n$ is still invertible mod n with overwhelming probability. In particular, for each one of the values a_1, a_2, b generated by the signature card $\mathcal{F}_{SC}^{C,CL_2^*}$, their distribution is statistically indistinguishable

from the uniform distribution on QR_n, and thus their inverses mod n exist with overwhelming probability. Moreover, also the distribution of these inverses is statistically indistinguishable from the uniform distribution on QR_n.

Next, we point out that the distribution of values C_v and C_ω is statistically close from the random distribution on $QR_n \times QR_n$. This is simple to see, and we only give here a rather informal argument. Indeed, a_1 is a generator of QR_n with overwhelming probability. Since ω is a random integer of length l_n, then a_1^ω is almost uniformly distributed in QR_n. On the other hand, the value v used for the commitment C_v is obtained from the equation $v^e = {a_1}^{m_1} \cdot {a_2}^{m_2} \cdot b^s \cdot c \bmod n$ where all bases are values in QR_n. Hence, $v \in QR_n$ and is also invertible mod n with overwhelming probability. The same reasoning applies for the commitment C_ω, only that this time we use the fact that a_2 is a generator and that r_ω is a random value of length l_n. Since ω and r_ω are independent random variables, we can now conclude that the statement made in the beginning of the paragraph is true.

What is left to show now is how to prove that PK is a statistical zero-knowledge proof. We do this in two steps: first we show that PK is a statistical honest verifier zero-knowledge proof. For the second step we will use some results from [CD98].

In order to prove the zero-knowledge honest verifier property for PK we have to construct a simulator *Sim* which having black box access to a copy of the honest recipient R_2, can simulate in a statistically indistinguishable way the view of R_2 when interacting with the honest sender C. The simulator *Sim* performs the following actions: it chooses three values e'_1, e'_2, $e'_3 \xleftarrow{r} [0, 2^k)$, in the same way as the honest R_2 does and computes the respective commitments c_1, c_1 and c_3. Then it simulates the first step of the prover as follows. *Sim* takes three random values z_1, z_2, $z_3 \xleftarrow{r} [0, 2^k)$. Since we assume the case of honest verifier R_2, we have that with overwhelming probability the unveil messages of the commitments c_1, c_1 and c_3 equal to the initial messages e'_1, e'_2, e'_3. Thus *Sim* simply computes $e_1 = e'_1 + z_1 \bmod 2^k$, $e_2 = e'_2 + z_2 \bmod 2^k$ and $e_3 = e'_3 + z_3 \bmod 2^k$. *Sim* chooses next eight values s_1 to s_8 at random from the following intervals:

- $s_1 \in [0, 2^{l_e} - 1]$
- $s_7 \in [0, 2^{l_m} - 1]$
- $s_8 \in [0, 2^{l_n} - 1]$
- $s_2 \in [0, 2^{l_s} \cdot 2^{2 \cdot k} + 2^{l_s} \cdot 2^k - 2^k - 1]$
- $s_3,\ s_6 \in [0, 2^{l_e} \cdot 2^{l_n} \cdot 2^{2 \cdot k} + 2^{l_e} \cdot 2^{l_n} \cdot 2^k - 2^k - 1]$
- $s_4,\ s_5 \in [0, 2^{l_n} \cdot 2^{2 \cdot k} + 2^{l_n} \cdot 2^k - 2^k - 1]$

Next *Sim* computes also:

- $d_1 = C_v^{s_1} \cdot \left(\frac{1}{b}\right)^{s_2} \cdot \left(\frac{1}{a_1}\right)^{s_3} \cdot \left(\frac{1}{COM \cdot c}\right)^{e_1} \bmod n,$
- $d_2 = a_1^{s_4} \cdot a_2^{s_5} \cdot \left(\frac{1}{C_\omega}\right)^{e_2} \bmod n$
- $d_3 = C_\omega^{s_1} \cdot \left(\frac{1}{a_1}\right)^{s_3} \cdot \left(\frac{1}{a_2}\right)^{s_6} \bmod n$
- $d_4 = a_1^{s_7} \cdot a_2^{s_8} \cdot \left(\frac{1}{COM}\right)^{e_3} \bmod n$

We argue that this simulation made by *Sim* is statistically indistinguishable from the view of the honest recipient R_2 when interacting with C for the proof PK. The reason is as follows. First, the values e_1, e_2 and e_3 computed by *Sim* have exactly the same distribution as the corresponding values obtained from an interaction between C and an honest R_2. The values s_i, for $i \in \{1, \ldots, 8\}$ are also identically distributed as their counterparts obtained in a proof PK.

Finally, the distribution of each of the values d_1, d_2, d_3 and d_4, as calculated by *Sim*, is statistically close to the uniform distribution on QR_n. This holds as the bases for all exponentiations *Sim* uses to compute d_1, d_2, d_3 and d_4 are, with overwhelming probability, generators of QR_n. Also the length of n (and thus the length of $\phi(n)$) is negligible compared to the length of every exponent used in the above-mentioned calculations.

Similarly, in an interaction between C_2 and an honest R_2 the values d_1, d_2, d_3 and d_4 are almost uniformly distributed in QR_n. Thus, we have shown that PK is a statistical honest-verifier zero-knowledge proof. The next step is to prove that the zero-knowledge property of the proof PK holds with respect to every PPT verifier R_2'. For this we invoke a result from [DN02] which states that if a proof PK is statistically honest-verifier zero-knowledge and the verifier commits to its challenges by using an unconditionally binding commitment scheme, which is the case of our construction, then PK is actually a statistical zero-knowledge proof. By showing that PK is statistical zero-knowledge proof, we have actually obtained that the view of $\mathcal{Z}$ when interacting with C_2 and the adversary $\mathcal{A}$ in the real world and the view of $\mathcal{Z}$ when interacting with $\mathcal{S}$ in the ideal world are statistically indistinguishable.

Finally, we investigate the view of the environment $\mathcal{Z}$ when interacting with C_3 and $\mathcal{A}$ in the real world on the one hand, and $\mathcal{S}$ and $\mathcal{F}_{COM}^{C,R}$, in the ideal world, on the other hand. In the real world interaction, the environment $\mathcal{Z}$ receives through $\mathcal{A}$ the committed value α and the unveil information β for the commitment COM. The value α was sent by the environment to C and the commitment value has been received by $\mathcal{Z}$ earlier. In order to have the two views statistically indistinguishable, the ideal simulator $\mathcal{S}$ has to open the commitment COM' to values α and β' such that $COM' = {a_1}^{\alpha} \cdot {a_2}^{\beta'} \bmod n$ and the distribution of β' is statistically close to the distribution of β from the real world. If $\mathcal{S}$ fails to do this only with negligible probability, then it is clear that also for this part of the protocol the two views of the environment $\mathcal{Z}$ are statistically indistinguishable.

By construction, $\mathcal{S}$ simulates the ideal functionality $\mathcal{F}_{SC}^{C,CL_2^*}$, thus it learns the value d chosen by the key generation algorithm of the signature scheme. The following property is fulfilled by d:

$$a_1 = {a_2}^{d} \bmod n$$

The value COM' has to satisfy the two equations listed below. The first one is by construction and the second one is needed in order to obtain the correct unveil information for value α and commitment COM':

$$COM' = {a_1}^{0} \cdot {a_2}^{\gamma} \bmod n \tag{5.1}$$

and

$$COM' = {a_1}^{\alpha} \cdot {a_2}^{\beta'} \bmod n \tag{5.2}$$

By combining the relations and rewriting everything to the base a_2 we obtain:

$${a_2}^{\gamma} = {a_2}^{d \cdot \alpha + \beta'} \bmod n$$

This is an equation in the variable β' and it has a very simple solution:

$$\beta' = \gamma - d \cdot \alpha \bmod \phi(n) \tag{5.3}$$

All values γ, d, and α are known to the ideal simulator $\mathcal{S}$ as well as the factorization of n. This enables $\mathcal{S}$ to compute also $\phi(n)$ and finally calculate the β' as described by the relation (5.3). Let $D_\beta = \{\beta : \beta \xleftarrow{r} \mathbb{Z}_n\}$ be the distribution of value β used in commitment COM. This is actually the uniform distribution on $\mathbb{Z}_n$. Let $D_{\beta'} = \{\beta' : d \xleftarrow{r} \mathbb{Z}_{\phi(n)}, \gamma \xleftarrow{r} \mathbb{Z}_n, \beta' = \gamma - d \cdot \alpha \bmod \phi(n)\}$ be the distribution of value β' that the ideal simulator $\mathcal{S}$ sends to the simulated $\mathcal{A}$. In order to complete the proof, it remains to be shown that D_β and $D_{\beta'}$ are statistically indistinguishable.

First we argue that the uniform distributions on $\mathbb{Z}_n$ and $\mathbb{Z}_{\phi(n)}$ are statistically indistinguishable. Indeed, since n is a special RSA modulus, $\phi(n) = (p-1)\cdot(q-1)$. The ratio of number of integers in $\mathbb{Z}_{\phi(n)}$ and $\mathbb{Z}_n$ is $\frac{\phi(n)}{n} = 1 - \frac{1}{p} - \frac{1}{q} + \frac{1}{p\cdot q}$. Since both p and q are integers of length k the above ratio differs from 1 only by a negligible function in k. Thus, the uniform distributions on the sets $\mathbb{Z}_{\phi(n)}$ and $\mathbb{Z}_n$ are statistically indistinguishable.

Next we show that $D_{\beta'}$ is statistically indistinguishable from the random distribution on $\mathbb{Z}_{\phi(n)}$. From the statement proved above combined with the fact that γ is a random value in $\mathbb{Z}_n$ we have that the distribution of γ differs only with negligible probability from the uniform distribution on $\mathbb{Z}_{\phi(n)}$. By construction, d is uniformly distributed in $\mathbb{Z}_{\phi(n)}$ and if α is invertible mod $\phi(n)$ then also the product $d \cdot \alpha$ is uniformly distributed in $\mathbb{Z}_{\phi(n)}$. But the fact that an element of $\mathbb{Z}_{\phi(n)}$ is invertible is not fulfilled only with negligible probability in the length of n. So we have proved also the second statement.

Combining the two claims made above, we obtain that D_β and $D_{\beta'}$ are statistically indistinguishable. This was the last step needed to show that in the case of a corrupted verifier for the protocol π, the views of the environment in the real and the ideal world are statistically indistinguishable. □

Together with the argumentation for the case of the corrupted sender, we have shown that protocol π long-term UC emulates functionality $\mathcal{F}_{COM}^{C,R}$ and thus conclude the proof of theorem 30. □

We turn our attention now towards constructing a commitment protocol that is long-term UC secure in the $\mathcal{F}_{CL_2}^{C,SC}$-hybrid model. The first motivation to do this is to build an efficient and secure commitment which uses an already existing signature scheme, in this case CL_2. The second motivation is that we present a way for constructing a long-term secure UC commitment from the protocol we have designed by using the composition theorem.

In the first step we show that the functionality $\mathcal{F}_{SC}^{C,CL_2^*}$ long-term UC emulates the ideal functionality $\mathcal{F}_{CL_2}^{C,SC}$. This will be proved in the next lemma. Then, we use the composition theorem and obtain that $\pi^{\mathcal{F}_{CL_2}^{C,SC} \setminus \mathcal{F}_{CL_2^*}^{C,SC}}$ long-term UC emulates π. By π we denote the protocol described in theorem 30. Finally, adding the result of the theorem 30 we obtain a long-term UC secure commitment in the $\mathcal{F}_{CL_2}^{C,SC}$-hybrid model. This result will be proved in theorem 34.

Lemma 33 (Long-term UC Emulation of $\mathcal{F}_{CL_2^*}^{C,SC}$ by $\mathcal{F}_{CL_2}^{C,SC}$)**.** *The ideal functionality $\mathcal{F}_{CL_2}^{C,SC}$ long-term UC emulates $\mathcal{F}_{CL_2^*}^{C,SC}$.*

Proof. According to the definition of emulation in the long-term UC model, we have to show that for every PPT adversary $\mathcal{A}$ interacting with $\mathcal{F}_{CL_2}^{C,SC}$ there exist a PPT adversary $\mathcal{S}$ interacting with $\mathcal{F}_{CL_2^*}^{C,SC}$ such that for every environment $\mathcal{Z}$ the two views he obtains from interacting with $\mathcal{A}$ and $\mathcal{F}_{CL_2}^{C,SC}$ on one hand and with $\mathcal{S}$ and $\mathcal{F}_{CL_2^*}^{C,SC}$ on the other hand, are statistically indistinguishable. We let adversary $\mathcal{S}$ run a simulated copy of the real adversary $\mathcal{A}$ which he will connect to $\mathcal{Z}$ and to the functionality $\mathcal{F}_{CL_2^*}^{C,SC}$. The only difference between the two protocols is in the way

the value a_1 is chosen by the two signature schemes CL_2 and CL_2^*, respectively. As seen in the proof for lemma 29, the two distributions for the value a_1 in the two schemes are identical if the value a_2 chosen by CL_2^* fulfils the property $\langle a_2 \rangle = QR_n$. From the result mentioned in the same lemma, it is also known that the probability for a_2 of not fulfilling the above mentioned property is negligible. Thus it becomes clear that the two views of $\mathcal{Z}$ are statistically indistinguishable and this ends the proof. □

The final theorem in this chapter shows that it is possible to implement an efficient long-term UC commitment also by using a signature card with the signature scheme CL_2.

Theorem 34 (Efficient Long-term Secure UC Commitment using $\mathcal{F}_{CL_2}^{C,SC}$)**.** *Assume that the strong RSA assumption holds. Then there is a nontrivial protocol ρ that long-term UC realizes $\mathcal{F}_{COM}^{C,R}$ and uses one instance of an ideal functionality $\mathcal{F}_{CL_2}^{C,SC}$.*

Proof. Let π be the protocol designed in the proof of theorem 30. From lemma 33 we have that $\mathcal{F}_{CL_2}^{C,SC}$ long-term UC emulates $\mathcal{F}_{CL_2^*}^{C,SC}$. Since π uses an instance of $\mathcal{F}_{CL_2^*}^{C,SC}$ and by the use of the composition theorem for the long-term UC model, we obtain that $\pi^{\mathcal{F}_{CL_2}^{C,SC} \setminus \mathcal{F}_{CL_2^*}^{C,SC}}$ long-term UC emulates π. But, by theorem 30, the protocol π long-term UC emulates the ideal functionality $\mathcal{F}_{COM}^{C,R}$. Since it is easy to see that long-term UC emulation is a transitive property, we obtain that $\pi^{\mathcal{F}_{CL_2}^{C,SC} \setminus \mathcal{F}_{CL_2^*}^{C,SC}}$ long-term UC emulates $\mathcal{F}_{COM}^{C,R}$ and thus finish our proof. □

To conclude, in this chapter we have shown how to design an efficient long-term UC secure protocol and then prove its security. This is the protocol π detailed in the proof of theorem 30. Its efficiency stems from the fact that it requires a number of only thirteen exponentiations from the sender and eleven exponentiations from the recipient. This was possible by using a signature card and efficient zero-knowledge proofs of knowledge modulo a special RSA n.

Chapter 6

Conclusions

We have examined the problem of designing long-term universally composable secure commitment protocols that have practical efficiency and can be build using already existing hardware. A commitment protocol that fulfills this strong security requirement has been previously constructed [MQU07]. It makes use of a device available on the market: the signature card. Still, the emphasis of this construction was on proving the existence of such commitments under general cryptographic assumptions, like the existence of one-way functions, rather than on efficient implementation.

The construction we have proposed builds on the idea from [MQU07] of using a signature card. One improvement is that our protocol needs to query such a card only once. Previously, the signature card had to be queried twice. The reason why such a query is particularly expensive for this protocol is the following. In order to maintain long-term UC security, every time the sender wants to use a signature on a message signed by its card, it cannot send the signature in plain, but it needs to make a zero-knowledge proof of knowledge of this signature. We have reduced the number of queries made by the sender, to its signature card, by slightly modifying the signature scheme from [CL02] and giving it a trapdoor property. Another reason for choosing this signature scheme is that it allows us to use the efficient zero-knowledge protocols from [CL02] for constructing our long-term UC commitment. In this way we obtain a six-round commitment protocol which requires only thirteen exponentiations for the sender and eleven exponentiations for the recipient.

For this protocol, the recipient had to commit to three random values using an unconditionally binding commitment scheme. As future work, we want to investigate if the proposed protocol remains long-term UC secure without the use of these commitments. Proving this would consequently reduce the number of rounds for our long-term UC protocol to four. Based on our current work, a related topic to be further investigated is the possibility of building a scheme that allows for multiple messages to be committed simultaneously but unveiled individually. A slightly different direction is to apply the techniques used here in order to construct a long-term UC secure zero-knowledge protocol.

Bibliography

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BDD+06] Michael Backes, Anupam Datta, Ante Derek, John C. Mitchell, Ajith Ramanathan, and Andre Scedrov. Games and the impossibility of realizable ideal functionality. Workshop on Formal and Computational Cryptography (FCC'06), affiliated with ICALP'06, 2006.

[BPW04] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In *TCC*, pages 336–354, 2004.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *IEEE Symposium on Foundations of Computer Science*, pages 136–145, 2001.

[Can05] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic or can zero-knowledge be for free? In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 424–441. Springer-Verlag, 1998.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology CRYPTO '01*, pages 19–40, 2001.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks (SCN)*, volume 2576 of *Lecture Notes in Computer Science*, pages 268–289. Springer Verlag, 2002.

[CLOS02] Ran Canetti, Yehuda Lindell, RAfail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Symposium of Theory of Computing (STOC)*, pages 494–503, 2002.

[CM97] Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In *CRYPTO*, pages 292–306, 1997.

[DF02] Ivan Damgård and Eiichiro Fujisaki. A statistically-hiding integer commitment scheme based on groups with hidden order. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pages 125–142. Springer-Verlag, 2002.

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 581–596. Springer-Verlag, 2002.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th Symposium of Theory of Computing (STOC)*, pages 409–418, 1998.

[Fei92] Uriel Feige. *Alternative Models For Zero Knowledge Interactive Proofs*. PhD thesis, Weizmann Institute of Science, 1992.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[Gol01] Oded Goldreich. *Foundations of Cryptography - Volume 1 (Basic Tools)*. Cambridge University Press, 2001.

[MQU07] Jörn Müller-Quade and Dominique Unruh. Long-term security and universal composability. In *Theory of Cryptography, Proceedings of TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 41–60. Springer Verlag, 2007.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for NP using any one-way permutation. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 11(2):87–108, 1998.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 245–254. ACM, 2000.