U N I V E R S I T Y OF T A R T U
Faculty of Mathematics and Computer Science
Institute of Computer Science

Liina Kamm

Homological Classification of Commitment Schemes

Master's Thesis

Supervisor: Ahto Buldas, PhD
Instructor: Sven Laur, MSc

TARTU 2007

Contents

1 Introduction

Mankind has always had secrets and with secrets comes the need to hide them and transfer them to trusted parties without the enemy finding out what is being sent. The original aim of cryptography was to provide secrecy and various "unbreakable" codes were designed already in ancient Greece. During the second half of the 20th century cryptography matured, as computers started playing a bigger role in its applications. Secrecy ceased being the only goal and was soon accompanied by integrity, fairness and anonymity. Also, a sound theoretical base was established for cryptography. Contemporary cryptography is based on complexity theory and most of its protocols rely on the existence of computationally hard problems. Such an approach allows to make the assumptions very clear, widely acceptable and testable in practice, e.g., the RSA breaking challenge. Some of the main primitives of cryptography are encryption and commitment schemes, zero knowledge proofs, oblivious transfer, authentication, identification and pseudorandom bit generation. In this master's thesis, we concentrate on commitment schemes and their relationship with encryption schemes.

Commitment schemes are a means of temporally hiding secret information so that it is verifiable in spite of the possible bias from the sender or the receiver. These schemes are an important building block for different cryptographic protocols and are used, for example, in digital timestamping, secret sharing schemes, zero knowledge proofs, electronic voting and secure multiparty computation.

In a sense, commitment schemes are in a central place in cryptography, as they form an essential part of several complex protocols. We can informally look at one of the most common applications of commitment schemes: a fair coin-flipping [Blu81]. This protocol for sharing random results is the heart of secure multiparty computation. Together with zero knowledge, essentially any complex cryptographic protocol needs fair cointosses. Consider the following example: we have two parties Alice and Bob who wish to determine the winner of a cointoss. Unfortunately Alice and Bob are at different locations and have to do this without both of them being able to see the outcome. Alice flips the coin, but Bob cannot entirely trust that Alice will tell him the right result once he has revealed his choice of heads or tails. So Alice puts the result in a box, locks it and sends it to Bob. Now Bob can reveal whether he chose heads or tails. Alice then sends him the key to the box so he can be sure of the result. This process can be repeated as many times as necessary to get the needed amount of random values and it has been widely used in the constructions of many cryptographic protocols [Gol04].

Unfortunately, so far there has been no concise overview of commitment schemes in widespread cryptographic literature. The objective of this thesis is to systematise different properties and examples of commitment schemes. We show how commitment schemes are in correspondence with encryption schemes and how

properties of both notions are related to each other. We give proofs to different kinds of properties and relationships, thus, explicitly demonstrating different proving techniques that are often used in cryptography.

We give a short overview of the different properties of commitment schemes discussed in this master's thesis.

- In Chapter 2 we give definitions and descriptions for the notions used later in the thesis. We talk about computationally hard problems as important building blocks for constructing cryptographic protocols.

- In Chapter 3 we show which functions are necessary in a commitment scheme and we describe its two basic properties: hiding and binding. To illustrate the different combinations of properties, we describe two simple commitment schemes.

- In Chapter 4 we discuss the duality between encryption and commitment schemes. We give a definition for extractability—a property that is necessary for the existence of a canonical correspondence between encryptions and commitments.

- Chapter 5 deals with simulatable commitments and equivocability—the property that is necessary for achieving simulatability of cryptographic protocols. This property is essential for simple proofs of secure multiparty computation in the malicious model, where an adversary can deviate from the protocol. We also describe two commitment schemes that are equivocable.

- Finally, in Chapter 6, we come to the non-malleability property of commitment schemes needed for key-agreement and authentication protocols where the communication is vulnerable to attacks from malicious parties. We show how it relates to properties of encryption schemes and we give an example of a provably non-malleable commitment scheme.

2 Preliminaries

2.1 Elements of Statistics and Probability Theory

The probability of an event shows how the number of occurrences of this event relates to other events. In cryptography, the classical definition of probability is often sufficient, in particular, if we consider randomised algorithms that throw coins or choose elements uniformly from sets. By $\Pr[AB]$ we denote the probability that the events A and B will both occur. Conditional probability $\Pr[A|B]$ is the chance that an event A will occur on the condition that the event B has already taken place and

$$\Pr[A|B] = \frac{\Pr[AB]}{\Pr[B]} \ .$$

Let the set of elementary events Ω be divided into disjoint subsets $B_1, \ldots, B_n$ such that $B_1 \cup \ldots \cup B_n = \Omega$ and $B_i \cap B_j = \emptyset$ for $i \neq j$. Let A be an event in Ω, then the probability of the occurrence of A is computed in the following way

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|B_i] \cdot \Pr[B_i] \ .$$

In the same setting, the probability of the event B_j happening given the occurrence of A can be computed by using the Bayes' rule

$$\Pr[B_j|A] = \frac{\Pr[B_j] \cdot \Pr[A|B_j]}{\sum_{i=1}^{n} \Pr[B_i] \cdot \Pr[A|B_i]} \ .$$

Statistical difference shows how similar two given probability distributions are. Let D be a discrete space and let X and Y be probability distributions (or random variables) on D. The statistical difference between X and Y is

$$\|X - Y\| = \max_{S \subset D} |\Pr[X \in S] - \Pr[Y \in S]| \ .$$

No algorithm can distinguish two distributions with an advantage greater than the statistical difference. We give a lemma that gives an upper bound for statistical difference between $\mathbb{Z}_u$ and the set created by taking values from $\mathbb{Z}_T$ modulo u. We will need it later in the security proof for the Fujisaki-Okamoto commitment scheme in Section 5.2.2.

Lemma 2.1. *Let $T > u$ be integers and let $\mathcal{D} = \{v : v = w \mod u, w \in \mathbb{Z}_T\}$, then the statistical difference between $\mathcal{D}$ and $\mathbb{Z}_u$ is $\frac{u}{T}$.*

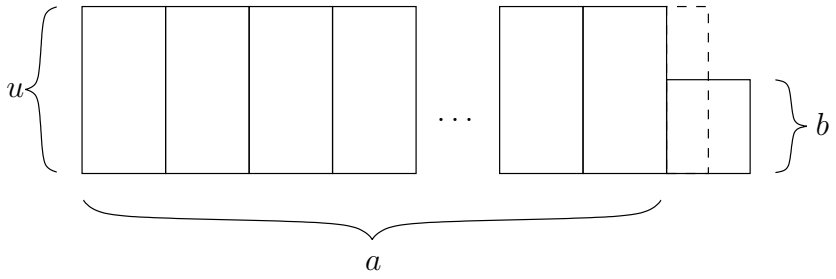

Figure 1: The relationship between the distributions of $\mathbb{Z}_u$ and $\mathcal{D}$

Proof. We define two values a and b to simplify our further computations. Let $a = \lfloor \frac{T}{u} \rfloor$, $b = T - au$ (Figure 1) and let y be an element chosen uniformly from $\mathbb{Z}_T$, we divide the elements of $\mathbb{Z}_u$ into two sets

$$T_0 = \{c \in \mathbb{Z}_u : \Pr[y \mod u = c]\} = \frac{a}{T} \ ,$$
$$T_1 = \{c \in \mathbb{Z}_u : \Pr[y \mod u = c]\} = \frac{a+1}{T} \ .$$

We notice that

$$|T_0| = u - |T_1| = u - b$$

and

$$|T_1| = T - au = b \ .$$

The statistical difference between $\mathbb{Z}_u$ and $\mathcal{D}$ can be expressed as

$$\varepsilon = \frac{|T_0|}{2} \cdot \left(\frac{1}{u} - \frac{a}{T} \right) + \frac{|T_1|}{2} \cdot \left(\frac{a+1}{T} - \frac{1}{u} \right) = \frac{b(u-b)}{uT} \leq \frac{u}{T} \ .$$

□

2.2 Computationally Hard Problems

Many cryptographic protocols, also commitment schemes, depend on different hard problems that cannot be efficiently solved for an arbitrary input. The security proofs are given in terms of the hardness of the underlying problem—as long as it is very difficult to solve the problem, the protocol is secure. This section gives an overview of the computationally hard problems that are the basis for different cryptographic protocols.

2.2.1 Hard Problems in Finite Groups

Finite groups are often used in cryptography as a source of computationally hard problems. We take a look at three problems based on the assumption that is is difficult to compute the discrete logarithm in some groups. This was first mentioned in the article [DH76] and is one of the main difficult problems that cryptographic protocols rely on.

Before getting to the three problems, we will look at some properties of finite groups and their elements. A group $\mathbb{G}$ is *cyclic* if there exists a *generator* $g \in \mathbb{G}$ so that for every element $y \in \mathbb{G}$ there exists an integer i so that $y = g^i$. The *order* of an element $y \in \mathbb{G}$ is the smallest integer i so that $y^i = 1$. The order of a cyclic group is the order of its generator and also the number of elements in the group. Powers of any group element g of order n form a subgroup:

$$\langle g \rangle = \{g^i : 0 \leqslant i \leqslant n-1\} \ .$$

It is easy to see that $\langle g \rangle$ is a cyclic subgroup of $\mathbb{G}$ of order n, where n is also the order of g. Every cyclic group $\mathbb{G}_n = \langle g \rangle$ is isomorphic with $\mathbb{Z}_n$. The isomorphism translates multiplication into addition, as can be seen in Figure 2. This isomorphism is realized by the discrete logarithm function $\log : \mathbb{G}_n \to \mathbb{Z}_n$, where

$$\log(g^i) = i \mod n \ .$$

Note that $\log(ab) = \log(a) + \log(b)$. The inverse function is exponentiation $\exp : \mathbb{Z}_n \to \mathbb{G}_n$, where

$$\exp(i) = g^i \ .$$

One can, of course, define log and exp with respect to different bases. However, in this thesis we explicitly assume that they are taken with respect to base g. Next, we give definitions of three problems in the group $\mathbb{G}_n$ and then discuss them, and their relations to each other, less formally.

Definition 2.1. *A group $\mathbb{G}_n$ is a (t, ε)-*Discrete Logarithm group *if for any t-time adversary A:*

$$\Pr\left[x \leftarrow \mathbb{Z}_n, y \leftarrow g^x : A(g, y) = x\right] \leqslant \varepsilon \ . \tag{1}$$

Definition 2.2. *A group $\mathbb{G}_n$ is a (t, ε)-*Computational Diffie-Hellman group *if for any t-time adversary A:*

$$\Pr\left[a, b \leftarrow \mathbb{Z}_n : A(g, g^a, g^b) = g^{ab}\right] \leqslant \varepsilon \ . \tag{2}$$

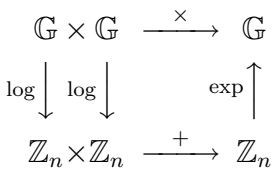

Figure 2: Isomorphism of $\mathbb{G}$ and $\mathbb{Z}_n$

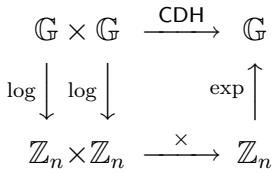

Figure 3: CDH problem and DL

Definition 2.3. *A group $\mathbb{G}_n$ is a* (t, ε)-Decisional Diffie-Hellman group *if for any t-time adversary A:*

$$\left| \Pr \left[\begin{matrix} a, b \leftarrow \mathbb{Z}_n : \\ A(g, g^a, g^b, g^{ab}) = 1 \end{matrix} \right] - \Pr \left[\begin{matrix} a, b, c \leftarrow \mathbb{Z}_n : \\ A(g, g^a, g^b, g^c) = 1 \end{matrix} \right] \right| \leqslant \varepsilon \ . \tag{3}$$

Less formally, the inequality (1) means that in a (t, ε)-discrete logarithm group it is difficult for an adversary to efficiently find the smallest non-negative integer x, $0 \leqslant x \leqslant n - 1$, so that $g^x = y$. Since

$$g^x = y \Longrightarrow x = \log y \ ,$$

we can simply say that on average it is difficult to find the discrete logarithm of an arbitrary element of the group.

The CDH problem states that the adversary is given three elements g, g^a and g^b from $\langle g \rangle$, where $a, b \leftarrow \{0, \ldots, n-1\}$. In a CDH-group (2) it is infeasible to compute the value g^{ab}. In the DDH problem, the adversary is given four elements g, g^a, g^b and y from $\langle g \rangle$, where $a, b \leftarrow \{0, \ldots, n-1\}$. The assumption (3) states that it is difficult to distinguish whether y is equal to g^{ab} or whether it is simply a random element of the group.

The CDH assumption is related to the discrete logarithm assumption. If computing discrete logarithms in $\mathbb{G}_n$ were easy, then the CDH assumption would be false, because given a tuple (g, g^a, g^b), the adversary A could find a or b and, thus, easily compute g^{ab}. This is illustrated in Figure 3 that shows how the CDH problem is related to the multiplication of exponents. However, it is an open problem to determine whether the discrete logarithm assumption is equivalent to CDH, though in certain special cases this can be shown to be the case [JN03].

The relationships between the three assumptions are given on Figure 4. The DDH assumption is related to the discrete logarithm assumption for the same reasons as the CDH assumption. It is believed that the DDH assumption is stronger than discrete logarithm, because there are groups for which detecting DDH tuples is easy but computing discrete logarithms is believed to be hard [Bon98]. The DDH and CDH assumptions are also related to each other. If computing g^{ab} from

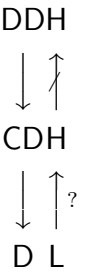

Figure 4: Relationship between the DL, CDH and DDH assumptions

(g, g^a, g^b) were easy, it would also be easy to detect DDH tuples. It is believed that DDH is a stronger assumption than CDH, because there are groups for which detecting DDH tuples is easy, but solving the CDH problem is believed to be hard [Bon98].

In the following lemma, we describe the *collision extraction property* of discrete logarithm. It is used in many of the algorithms for finding the discrete logarithm of an element.

Lemma 2.2. *In a group $\mathbb{G}_q = \langle g \rangle$, where q is a prime, it is possible to efficiently compute the discrete logarithm x of an element $y \in \mathbb{G}_q$ with respect to base g, if we know four values $m_0, m_1, r_0, r_1 \in \mathbb{Z}_q$ such that $g^{m_0} y^{r_0} = g^{m_1} y^{r_1}$ and $r_0 \neq r_1$.*

Proof. As $y = g^x$, we can do the following to compute the discrete logarithm of y

$$\begin{aligned} g^{m_0} g^{x r_0} &= g^{m_1} g^{x r_1} \\ g^x &= g^{\frac{m_0 - m_1}{r_1 - r_0}} \\ x &\equiv \frac{m_0 - m_1}{r_1 - r_0} \mod q \ . \end{aligned}$$

As $r_0 \neq r_1$ and q is a prime, we know that $(r_1 - r_0)^{-1}$ exists. □

Self-Reducibility. The discrete logarithm, CDH and DDH problems are random self-reducible. Intuitively, this means that if the adversary is able to efficiently compute the discrete logarithm on average, he is able to do it for almost all inputs. More formally, the worst case complexity is close to the average case complexity since the problem can be re-randomised.

To illustrate the concept of random self-reducibility, we present algorithms for re-randomising these problems. The basic idea of the algorithms is to construct a new adversary B that uses the old adversary on a variable input. In the discrete logarithm case the adversary B_{DL} gets g and y as input and has to output $\log_g y$.

The adversary re-randomises the element y by uniformly choosing $z \leftarrow \mathbb{Z}_n$ and giving the values g and yg^z as input to the adversary A_{DL}. The discrete logarithm can be computed from the output x' of A_{DL} by a simple subtraction $\log_g y = x' - z$. In brief, the reduction algorithm for the self-reducibility of the discrete logarithm can be written down as follows:

$B_{\mathrm{DL}}(g, y)$
$\left[\begin{array}{l} z \leftarrow \mathbb{Z}_n \\ y' \leftarrow yg^z \\ \mathsf{Return}\ A_{\mathrm{DL}}(g, y') - z \end{array}\right.$

It is quite straightforward to see that the algorithm is correct. Let $y = g^x$, then $y' = yg^z = g^x g^z = g^{x+z}$ and if the adversary A_{DL} succeeds in computing the discrete logarithm for y' then B_{DL} succeeds in computing $\log y$. The reduction also preserves the advantage of the adversary A_{DL}, as the adversary B_{DL} successfully outputs $\log y$ if A_{DL} correctly computes the $\log y'$:

$$\begin{aligned} \Pr\left[B(g, y) = x\right] &= \Pr\left[x' \leftarrow \mathbb{Z}_n, y' \leftarrow g^{x'} : A_{\mathrm{DL}}(g, y') = x'\right] \\ &= \Pr\left[x \leftarrow \mathbb{Z}_n, y \leftarrow g^x : A_{\mathrm{DL}}(g, y) = x\right] = \mathrm{Adv}(A) \ . \end{aligned}$$

The adversary B_{CDH} for the CDH problem is given g, g^a and g^b and has to compute g^{ab}. The re-randomising is done in much the same way as with the discrete logarithm problem. Two elements are uniformly chosen $u, v \leftarrow \mathbb{Z}_n$ and the adversary A_{CDH} is given g, $g^a g^u$ and $g^b g^v$ as input. The value g^{ab} can be computed by dividing the output $g^{a'b'}$ of A_{CDH} with the product $(g^a)^v (g^b)^u g^{uv}$. The reduction algorithm for the self-reducibility of the computational Diffie-Hellman assumption can compactly be written in the following way:

$B_{\mathrm{CDH}}(g, g^a, g^b)$
$\left[\begin{array}{l} u, v \leftarrow \mathbb{Z}_n \\ g^{a'} \leftarrow g^a g^u \\ g^{b'} \leftarrow g^b g^v \\ z \leftarrow A_{\mathrm{CDH}}(g, g^{a'}, g^{b'}) \\ \mathsf{Return}\ z(g^a)^{-v}(g^b)^{-u} g^{-uv} \end{array}\right.$

It is easy to see that the algorithm is correct. We know that $g^{a'} = g^a g^u = g^{a+u}$ and $g^{b'} = g^b g^v = g^{b+v}$. Hence, if the adversary A outputs $g^{a'b'}$, then

$$g^{a'b'} = g^{(a+u)(b+v)} = g^{ab+av+ub+uv} = g^{ab}(g^a)^v (g^b)^u g^{uv} \ .$$

Now, if the adversary A_{CDH} succeeds in computing $g^{a'b'}$ then B_{CDH} succeeds in computing $g^{ab} = g^{a'b'}(g^a)^{-v}(g^b)^{-u}g^{-uv}$. Similarly to the discrete logarithm problem, the reduction preserves the advantage of the adversary A_{CDH}, meaning that the adversary B_{CDH} is successful in computing g^{ab} if A_{CDH} can compute $g^{a'b'}$ given a tuple $(g, g^{a'}, g^{b'})$:

$$\begin{aligned}\Pr\left[B(g, g^a, g^b) = g^{ab}\right] &= \Pr\left[\begin{array}{l} u, v \leftarrow \mathbb{Z}_n, g^{a'} \leftarrow g^a g^u, g^{b'} \leftarrow g^b g^v : \\ A_{\text{CDH}}(g, g^{a'}, g^{b'}) = g^{a'b'} \end{array}\right] \\ &= \Pr\left[a, b \leftarrow \mathbb{Z}_n : A_{\text{CDH}}(g, g^a, g^b) = g^{ab}\right] = \text{Adv}(A) \ .\end{aligned}$$

Finally, we look at the self-reducibility of the DDH problem. The adversary B_{DDH} receives the values g, g^a, g^b and y. The problem lies in distinguishing whether y is equal to g^{ab} or whether it is a uniformly chosen element of $\mathbb{G}_n$. The adversary outputs a guess $i \in \{0, 1\}$, where $i = 1$, if the fourth element in the tuple is equal to g^{ab}, and $i = 0$ otherwise. The re-randomising algorithm uses some of the same logic as the CDH one. Two elements are uniformly chosen $u, v \leftarrow \mathbb{Z}_n$ and the values $g^{a'} \leftarrow g^a g^u$ and $g^{b'} \leftarrow g^b g^v$ are computed. In addition, we also need the value $y' = y(g^a)^v(g^b)^u g^{uv}$. The adversary A_{DDH} is given four values g, $g^{a'}$, $g^{b'}$ and y'. The adversary outputs a decision $i' \in \{0, 1\}$, indicating whether $y' = g^{a'b'}$. The adversary B_{DDH} gives this decision as the output. In short, the reduction algorithm for the self-reducibility of the decisional Diffie-Hellman is the following:

$B_{\text{DDH}}(g, g^a, g^b, y)$

$$\left[\begin{array}{l} u, v \leftarrow \mathbb{Z}_n \\ g^{a'} \leftarrow g^a g^u \\ g^{b'} \leftarrow g^b g^v \\ y' \leftarrow y(g^a)^v(g^b)^u g^{uv} \\ \textsf{Return } A_{\text{DDH}}(g, g^{a'}, g^{b'}, y') \end{array}\right.$$

It is quite simple to show that the algorithm is correct. We know that $g^{a'} = g^a g^u = g^{a+u}$ and $g^{b'} = g^b g^v = g^{b+v}$. If $y = g^{ab}$ then

$$y' = g^{ab}(g^a)^v(g^b)^u g^{uv} = g^{ab+av+ub+uv} = g^{(a+u)(b+v)} = g^{a'b'} \ .$$

If y chosen uniformly from $\mathbb{G}_n$, then y' is also has random distribution over the same group. Now if the adversary A_{DDH} succeeds in distinguishing $g^{a'b'}$ and an arbitrary element of $\mathbb{G}_n$ then B_{DDH} succeeds in distinguishing g^{ab} from an arbitrary element of $\mathbb{G}_n$. As before, the reduction preserves the advantage of the adversary A_{DDH}, meaning that the adversary B_{DDH} is successful if A_{DDH} can distinguish $g^{a'b'}$

from an arbitrary element $g^{c'}$ of $\mathbb{G}_n$. We know that

$$\Pr\begin{bmatrix} g^{a'} \leftarrow g^a g^u, g^{b'} \leftarrow g^b g^v, \\ g^{a'b'} \leftarrow g^{ab}(g^a)^v(g^b)^u g^{uv} : \\ A_{\mathrm{DDH}}(g, g^{a'}, g^{b'}, g^{a'b'}) = 1 \end{bmatrix} = \Pr\begin{bmatrix} a, b \leftarrow \mathbb{Z}_n : \\ A_{\mathrm{DDH}}(g, g^a, g^b, g^{ab}) = 1 \end{bmatrix}$$

and

$$\Pr\begin{bmatrix} g^{a'} \leftarrow g^a g^u, g^{b'} \leftarrow g^b g^v, \\ g^{c'} \leftarrow g^c(g^a)^v(g^b)^u g^{uv} : \\ A_{\mathrm{DDH}}(g, g^{a'}, g^{b'}, g^{c'}) = 1 \end{bmatrix} = \Pr\begin{bmatrix} a, b, c \leftarrow \mathbb{Z}_n : \\ A_{\mathrm{DDH}}(g, g^a, g^b, g^c) = 1 \end{bmatrix}$$

hold, hence, the probability that B_{DDH} outputs the right guess is equal to the probability that A_{DDH} succeeds in outputting the right guess.

2.2.2 Algorithms for the Discrete Logarithm Problem

The most obvious algorithm for finding the discrete logarithm is to compute $g^0, g^1, g^2, \ldots$ until the value y is reached. Let n be the order of g, then this method takes $\mathcal{O}(n)$ multiplications and is very inefficient if n is large.

Shanks' algorithm. This algorithm, also known as the baby-step giant-step algorithm, is a time-memory trade-off of the exhaustive search method. Let $y = g^x$ and let n be the order of g, then it is possible to write $x = im+j$, where $m = \lceil\sqrt{n}\rceil$, $0 \leqslant i, j \leqslant m$. This also means that

$$g^x = g^{im}g^j \implies y(g^{-m})^i = g^j \ .$$

The algorithm consists of the following steps:

1. For every j, $0 \leqslant j \leqslant m-1$, compute g^j;
2. Sort the computed pairs (j, g^j) according to the second element;
3. $z \leftarrow y$;
4. For every i, $0 \leqslant i \leqslant m-1$
   (a) Check whether z is the second component in some pair ,
   (b) If $z = g^j$, return $x = im + j$,
   (c) Otherwise set $z \leftarrow zg^{-m}$.

Shanks' algorithm is a standard meet-in-the-middle trick that is more effective than the naïve algorithm. It offers the complexity of $\mathcal{O}(\sqrt{n})$ group multiplications but requires the storage for $\mathcal{O}(\sqrt{n})$ group elements and the table takes $\mathcal{O}(\sqrt{n})$ multiplications to construct and $\mathcal{O}(n \log n)$ comparisons to sort [Sha71].

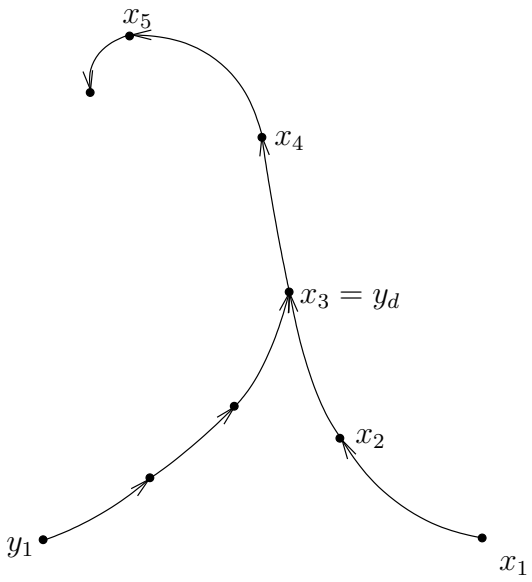

Figure 5: Pollard's lambda algorithm

Pollard's lambda algorithm. Pollard's lambda algorithm (Figure 5) is mainly used for finding discrete logarithms that lie in a short interval. This algorithm works on the collision extraction property of discrete logarithm. It is also known as the method of tame and wild kangaroos, because the main idea of the algorithm can be described as random walks of two kangaroos. The tame kangaroo jumps into the wild, digs a hole and waits for the wild kangaroo to fall into it. Jumping means exponentiation and if asked to find $\log y$ then one kangaroo starts from g and the other one from y. The algorithm is called the lambda algorithm for the λ shape of the kangaroos' paths [BSS99].

Pollard's rho algorithm. Pollard's rho (Figure 6) is a randomized algorithm with the same expected running time as Shanks' algorithm, but it requires a negligible amount of storage. For this reason, it is preferred when dealing with practical problems. It is a special case of the lambda algorithm and similarly to the latter, it works on the collision extraction property of discrete logarithm. The basic idea is to start walking and searching until a collision is found [MvOV01].

The Pohlig-Hellman algorithm. This algorithm takes advantage of the factorization of the order n of the group $\mathbb{G}_n$. Given the prime factorization of $n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$, $(e_1, \ldots, e_r \geqslant 1)$, the running time of the Pohlig-Hellman algorithm is $\mathcal{O}(\sum_{i=1}^{r} e_i(\lg n + \sqrt{p_i}))$ group multiplications. Basically, discrete logarithm of an element is found iteratively for every factor p_i. This implies, though that the

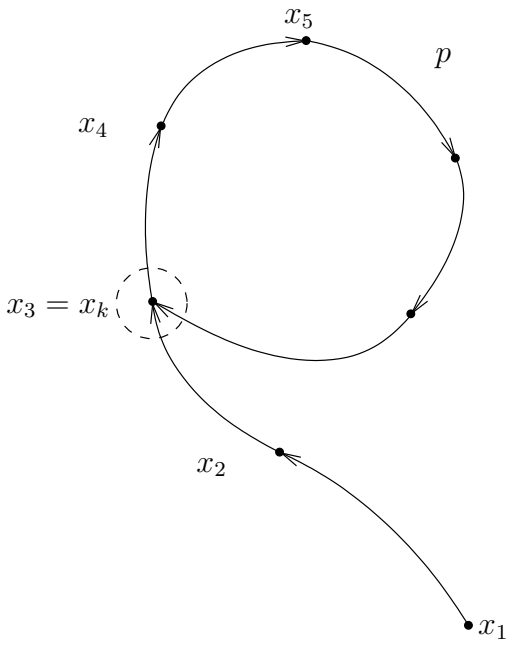

Figure 6: Pollard's rho algorithm

Pohlig-Hellman algorithm is efficient only if each prime divisor p_i of n is relatively small. If n is a prime, then the Pohlig-Hellman algorithm is the same as Shanks' algorithm [MvOV01].

The index-calculus algorithm. The algorithms described above are all generic—they work in any group. The index-calculus algorithm is the most powerful method known for computing discrete logarithms. This technique does not apply to all groups, but when it does, it often gives a subexponential-time algorithm. The two kinds of groups that this algorithm works on and that are used in practical applications are $\mathbb{Z}_p^*$ and $\mathbb{F}_{2^m}^*$ [MvOV01].

2.2.3 Factoring Problem and RSA Modulus

The *integer factoring problem* is to find for a given positive integer N its prime factorisation $N = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, where p_i are pairwise distinct primes and each $e_i \geqslant 1$. Finding out whether an integer is composite or prime is considered to be easier than the factoring problem. Therefore, when talking about algorithms for solving this problem, it is assumed that it has already been determined that N is composite. The efficiency of any factorisation algorithm depends on the size of the factors, especially the smallest one.

This problem is most difficult to solve, if $n = pq$, where p and q are large

primes. But even this might not be enough, as Pollard's $(p-1)$ algorithm works well, if the factors of $(p-1)$ are small, hence, it is easier to factorise n if the Eulerian function $\varphi(n)$ has small factors. A prime P is a safe prime if it is equal to $2p+1$, where p is also a prime. If n is the product of two safe primes $P = 2p+1$ and $Q = 2q+1$, then

$$\varphi(n) = \varphi(PQ) = (2p+1-1)(2q+1-1) = 2p2q = 4pq \ .$$

It is quite straightforward to see that if P and Q are large then p and q are also large and $\varphi(n)$ has large factors.

The first cryptoscheme to be built on this problem was the RSA encryption scheme [RSA78]. The RSA modulus n is a product of two large primes p and q of approximately the same size. As it is a publicly known value and the secret key is efficiently computable, if $\varphi(n)$ is known, then factorising n results in breaking the encryption. Therefore, it is necessary that n be a large value, e.g., 2048 bits long. More formally, an RSA modulus set $\mathsf{RSA\text{-}mod}(w)$ is a set that consists of pairs of safe primes (p,q) that are w bits long. The key generation algorithm Gen of the RSA scheme uniformly takes a pair (p,q) from the set and computes $n = pq$. We say that an RSA set is (t,ε)-secure if any t-time adversary A achieves advantage

$$\mathrm{Adv}(A) = \Pr\left[n \leftarrow \mathsf{Gen}, A(n) = p\right] \leq \varepsilon \ .$$

It is easy to sample an $\mathsf{RSA\text{-}mod}(w)$ set. It is estimated that a set is secure enough for all feasible computations if $\varepsilon = 2^{-80}$ and currently 1024 bit safe primes are considered sufficient.

Another interesting observation is that given the value of a multiple of $\varphi(n)$, we can also find the factors of n.

Lemma 2.3. *Let n be a product of two safe primes $P = 2p+1$ and $Q = 2q+1$. Then given $t\varphi(n)$, where $t \in \mathbb{N}$, it is possible to efficiently find P and Q.*

Proof. We know that $\varphi(n) = (P-1)(Q-1) = 2p2q = 4pq$ and we can express $t\varphi(n)$ as $2^s a$, where $a \in \mathbb{N}$ is odd. We know that $4pq | 4a$ and, hence we can say that $a = pqz$, where $z \in \mathbb{N}$ is odd. Next, we look at an element x to the power of $2a$. Let x be an arbitrary element of $\mathbb{Z}_n$, on one hand,

$$x^{2a} = (x^{2p})^{qz} \equiv 1^{qz} \equiv 1 \mod P$$

and on the other hand,

$$x^{2a} = (x^{2q})^{pz} \equiv 1^{pz} \equiv 1 \mod Q$$

then from the Chinese Remainder Theorem

$$x^{2a} \equiv 1 \mod n \ .$$

We define a group

$$\mathbb{G} = \{x \in \mathbb{Z}_n : x^a = \pm 1 \mod n\} \ ,$$

that is a multiplicative group in $\mathbb{Z}_n^*$. As a is odd, then $(-1)^a = -1^a$ and there exists an element x_0 such that $x_0 \equiv -1 \mod P$ and $x_0 \equiv 1 \mod Q$, and $x_0^a \neq \pm 1$, thus, the group $\mathbb{G}$ does not coincide with $\mathbb{Z}_n^*$ and $|\mathbb{G}| \leq \frac{|\mathbb{Z}_n^*|}{2}$. The element x_0 is from $\mathbb{Z}_n^* \backslash \mathbb{G}$ and the probability of finding such an element is not less than $\frac{1}{2}$, as $|\mathbb{Z}_n^* \backslash \mathbb{G}| \geq \frac{|\mathbb{Z}_n^*|}{2}$. One factor of n is $\gcd(x_0^a + 1, n)$, as x_0^a is equivalent with -1 either modulo P or Q. □

2.2.4 Algorithms for the Factoring Problem

The most obvious way to factor an integer n is by trial division. This algorithm divides n by all primes up to $\sqrt{n}$ and will completely factor the integer. This procedure takes roughly $\sqrt{n}$ divisions in the worst case when n is the product of two primes of approximately the same size. If all the factors are tested for primality then completely factoring n with this method takes $\mathcal{O}(p + \lg n)$ divisions, where p is the second largest prime factor of n. This method is considered useful for finding "small" prime factors quickly.

Some factoring algorithms are meant for factoring integers that have special properties. Pollard's rho algorithm, Pollard's $(p-1)$ algorithm, the elliptic curve algorithm, and the special number field sieve are special-purpose algorithms, the quadratic sieve and the general number field sieve are general-purpose algorithms. The running times of the general-purpose algorithms, depend on the size of n.

Pollard's rho algorithm is meant for finding small factors of a composite integer. The expected time to find a non-trivial factor of n is $\mathcal{O}(n^{\frac{1}{4}})$ [MvOV01].

Pollard's $(p-1)$ factoring algorithm can be used to efficiently find any prime factor p of n where all the prime factors of $(p-1)$ are not larger than a relatively small bound $B \in \mathbb{N}$, which is selected according to the amount of time one is willing to spend on this method before moving on to more general techniques. The value B is known as the smoothness bound. A number is considered B-smooth, if all of its prime factors are not larger than B. This, however, means that if p is a safe prime, this method is not very efficient and, hence, safe primes are usually required for a secure RSA modulus. The running time for finding this factor p is $\mathcal{O}(B \ln n / \ln B)$ modular multiplications [MvOV01].

In Pollard's $(p-1)$ algorithm $(p-1)$ is the order of the group $\mathbb{Z}_p^*$. The elliptic curve factoring algorithm is a generalisation of Pollard's $(p-1)$ algorithm in the sense that the group $\mathbb{Z}_p^*$ is replaced by a random elliptic curve group over $\mathbb{Z}_p$. If all the prime factors of the order of the group chosen are smaller than a pre-selected bound, the elliptic curve algorithm will find a non-trivial factor of n with high

probability. If not, the algorithm will probably fail but can then be repeated with a different choice of elliptic curve group. This algorithm has an expected running time of $\mathcal{O}(\exp((\sqrt{2}+o(1))(\ln p)^{\frac{1}{2}}(\ln\ln p)^{\frac{1}{2}}))$ [MvOV01].

The random square factoring methods attempt to find integers x and y at random so that $x^2 \equiv y^2 \mod n$. With probability at least $\frac{1}{2}$ it is the case that $x^2 \not\equiv \pm y^2 \mod n$ and so computing $\gcd(x-y,n)$ will give a non-trivial factor of n. The number field sieve factoring belongs to this family [MvOV01].

The number field sieve factoring is the fastest known factorisation algorithm. It has the expected running time of $\mathcal{O}(\exp((c+o(1))(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}))$, where $c = (64/9)^{\frac{1}{3}} \approx 1.923$, and it applies to all integers [MvOV01].

2.2.5 Pseudorandom Generators

When talking about different cryptographic notions, we often refer to generating a random value, however, true random is not easy to find in practice. A pseudorandom bit generator PRG [GGM84] provides a means for expanding a random bit sequence (the seed a) of length l into a bit sequence of length k. The output is not true random, because it is computed by PRG, moreover, providing the generator with the same seed more than once outputs the same value. In this thesis, when we talk about pseudorandom generators, we refer to pseudorandom bit generators, as these can also be used to generate uniformly distributed random numbers. These generators are important because they are fairly easy to construct and it is possible to build practically any symmetric protocols from them. A secure pseudorandom generator PRG expands the seed so that it is almost impossible to tell the difference between a real random sequence of k and the output of PRG.

Definition 2.4. *A pseudorandom generator* PRG *is a* (t,ε)-pseudorandom generator*, if a* t*-time adversary* A *achieves advantage*

$$\mathrm{Adv}(A) = \left| 2 \cdot \Pr\left[\begin{array}{l} s \leftarrow \{0,1\}, a \leftarrow \{0,1\}^l, y_1 \leftarrow \{0,1\}^k, \\ y_0 = \mathsf{PRG}(a) : A(y_s) = s \end{array} \right] - 1 \right| \leq \varepsilon \ .$$

Usually a one-way function, e.g., permutation, is used in the generator for compiling the output sequence. Among examples of pseudorandom generators are the Blum Blum Shub generator [BBS86], all stream ciphers (e.g. SNOW2.0 [EJ02]) and any strong block ciphers (e.g. AES [Nat01]) in counter mode.

2.2.6 Collision Resistance

Collision resistance is mainly talked about in the context of hash functions. Hash functions are among the most important tools for building cryptographic protocols. The hash function family $\mathcal{H} \subseteq \{h : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}\}$ converts a bit-string $m \in \mathcal{M}$ of

arbitrary finite length into a string $y \in \{0,1\}^n$, using a key $k \in \mathcal{K}$. Usually $\mathcal{M}$ is a sufficiently large subset of $\{0,1\}^t$, so it is possible, in practice, to hash a message of necessary length. We assume that $t > n$, thus, the function maps more than one value from $\{0,1\}^t$ to an arbitrary element of $\{0,1\}^n$ and therefore it is inevitable that collisions exist. This is the reason, why we talk about collision resistant not collision free hash function families. The unique association between the message and the hash can be at most computational.

The two main properties of hash functions are compression and ease of computation. In addition to these two properties, we give three more properties that are considered to be basic hash function properties. Let h be a hash function with inputs m and m' and output y. According to [MvOV01], preimage resistance or one-wayness means that it is computationally infeasible to find any input that hashes to a given output y for which an input is not yet known. Second preimage resistance or weak collision resistance implies that given y it is computationally infeasible to find an input $m' \neq m$, so that $h(m) = h(m')$. And finally, collision resistance or strong collision resistance means that it is computationally infeasible to find two inputs m and m' so that $m \neq m'$ and $h(m) = h(m')$.

Collision resistance is an important property of hash function families and it is essential for usage in cryptographic schemes. The formal definition of collision resistance together with many other properties is given in [RS04].

Definition 2.5. *A hash function family $\mathcal{H} = \{h : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}\}$ is* (t, ε)-collision resistant*, if any t-time adversary A has bounded advantage:*

$$\mathrm{Adv}^{\mathsf{Coll}}_{\mathcal{H}}(A) = \Pr\left[k \leftarrow \mathcal{K}, (m, m') \leftarrow A(k) : m \neq m', h_k(m) = h_k(m')\right] \leqslant \varepsilon \ .$$

It is simple to see that collision resistance implies second preimage resistance. Let the function family be collision resistant and let us fix the input value m. If the family does not have second preimage resistance, we are able to find another input value $m' \neq m$ so that $h(m) = h(m')$. But now we have found an input pair that hashes to the same value, thus contradicting collision resistance. Though collision resistance does not always imply preimage resistance, it does so for compressing function families and, thus, for most hash function families arising in practice, it is also reasonable to assume that it does [MvOV01].

In Section 3.1.2, we need universal hash functions. This property [CW77] is different from the previous three properties as it is an information theoretical property, i.e, it can resist unbounded attacks. Let $\mathcal{H} = \{h : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}\}$ be a hash function family. We say that $\mathcal{H}$ is a universal family of hash functions if for

$m_1, m_2 \in \mathcal{M}$ $(m_1 \neq m_2)$ and $t_1, t_2 \in \mathcal{Y}$, the following equations hold

$$\begin{aligned} \Pr\left[h \in \mathcal{H}, h(m_1) = t_1 \wedge h(m_2) = t_2\right] &= \frac{1}{|\mathcal{Y}|^2} \ , \\ \Pr\left[h \in \mathcal{H}, h(m_1) = t_1\right] &= \frac{1}{|\mathcal{Y}|} \ , \\ \Pr\left[h \in \mathcal{H}, h(m_1) = h(m_2)\right] &= \frac{1}{|\mathcal{Y}|} \ . \end{aligned}$$

For each $m_1 \neq m_2$, the values output by $h(m_1)$ and $h(m_2)$ are independent and of uniform distribution. An example of a universal hash function family is the following:

$$\mathcal{H} = \left\{h_{A,b} : A \in \{0,1\}^{nt}, b \in \{0,1\}^n\right\} \ ,$$

where $h_{A,b}(x)$ is defined as $h_{A,b}(x) = Ax + b$ [Sti06].

2.3 Game Based Security Definitions

The security of a cryptographic notion is often defined via game-playing—a security game between a challenger and an adversary is defined and the adversary has to win that game in order to succeed. There are different ways for expressing the advantages of adversaries in cryptographic games. Here we discuss two of them: comparison based and guessing based. In the comparison based game two worlds $\mathcal{G}_0$ and $\mathcal{G}_1$ are defined, the adversary is randomly put into one of these worlds and he has to output which world he is in. Let $\mathcal{G}_i^A$ be the result of the game with adversary A, then the advantage of the adversary is

$$\mathrm{Adv}_{\mathsf{dist}}(A) = \left|\Pr\left[\mathcal{G}_1^A = 1\right] - \Pr\left[\mathcal{G}_0^A = 1\right]\right| \ .$$

As there is no easily understandable interpretation for $\mathrm{Adv}_{\mathsf{dist}}$, it is better to look at a situation, where the adversary ends up in the world $\mathcal{G}_i$ with probability $\frac{1}{2}$ and see how well the adversary can determine which world he is in. Hence, the guessing based world begins with the challenger flipping a coin and choosing the world based on the result. The challenger begins playing the game with the adversary A in the chosen world and A has to guess the result of the cointoss based on the knowledge he gains during the game. The probability that he succeeds is

$$\Pr\left[i \leftarrow \{0,1\} : \mathcal{G}_i^A = i\right] \ .$$

Since in this case there is always a possibility of $\frac{1}{2}$ that the adversary simply flips a coin and guesses right, then we define the advantage of the adversary A as

$$\mathrm{Adv}_{\mathsf{guess}}(A) = \left|2 \cdot \Pr\left[i \leftarrow \{0,1\} : \mathcal{G}_i^A = i\right] - 1\right| \ .$$

It is intuitive that these two notions are connected. Consider the following equation

$$\begin{aligned}
\Pr\left[i \leftarrow \{0,1\} : \mathcal{G}_i^A = i\right] &= \frac{1}{2}\Pr\left[\mathcal{G}_0^A = 0\right] + \frac{1}{2}\Pr\left[\mathcal{G}_1^A = 1\right] \\
&= \frac{1}{2}(1 - \Pr\left[\mathcal{G}_0^A = 1\right]) + \frac{1}{2}\Pr\left[\mathcal{G}_1^A = 1\right] \\
&= \frac{1}{2} + \frac{1}{2}(\Pr\left[\mathcal{G}_1^A = 1\right] - \Pr\left[\mathcal{G}_0^A = 1\right]) \\
&= \frac{1}{2} + \frac{1}{2}\mathrm{Adv}_{\mathsf{dist}}(A) \ .
\end{aligned}$$

It is quite straightforward to see that if $\mathrm{Adv}_{\mathsf{comp}}(A) \leq \varepsilon$, then the advantage $\mathrm{Adv}_{\mathsf{guess}}(A)$ is not greater than ε either. On the other hand, let $\mathrm{Adv}_{\mathsf{guess}}(A) \leq \varepsilon'$ and consider another equation

$$\begin{aligned}
\left|\Pr\left[\mathcal{G}_1^A = 1\right] - \Pr\left[\mathcal{G}_0^A = 1\right]\right| &= \left|\Pr\left[\mathcal{G}_1^A = 1\right] - (1 - \Pr\left[\mathcal{G}_0^A = 0\right])\right| \\
&= 2 \cdot \left|\frac{1}{2}\Pr\left[\mathcal{G}_1^A = 1\right] - \frac{1}{2} + \frac{1}{2}\Pr\left[\mathcal{G}_0^A = 0\right]\right| \\
&= \left|2 \cdot \Pr\left[i \leftarrow \{0,1\} : \mathcal{G}_i^A = i\right] - 1\right| \leq 2\varepsilon' \ .
\end{aligned}$$

In this thesis we will use guessing based definitions, since we consider them more intuitive and easily manipulable. However, both of the choices are obviously equivalent as the same results can be obtained with comparison based definitions.

3 Basic Properties of Commitment Schemes

A commitment scheme is a means of showing another party that one knows a message m without initially revealing m. To think of a simple parallel to a commitment scheme, consider two parties *Alice* and *Bob*. *Alice* puts her message for *Bob* into a safe, seals it and sends it to *Bob* who cannot find out what is in the safe, but *Alice* cannot change it either. Later, when *Alice* is ready to reveal her secret, she sends the combination for the safe to *Bob* who can then open the safe and check the message. Commitment schemes have two essential properties: hiding and binding. Hiding, in this context, means that by looking at the safe, *Bob* cannot learn anything about the message inside, so this property ensures the safety of *Alice*. Binding, on the other hand, ensures the safety of *Bob* and means that *Alice* cannot send a false combination that opens a secret compartment in the safe and reveals the wrong message to *Bob*.

There are different ways for describing the functions of commitment schemes. In this thesis we follow the classical formalisation that has also been used in [Nao89, Cre02, DG03, LAN05]. A commitment scheme consists of three parts: key generation Gen, commitment Com and opening Open. The key generation algorithm generates the public parameters $\mathsf{pk} \leftarrow \mathsf{Gen}$ and we assume that this function is always run by a trusted third party. The commitment algorithm $\mathsf{Com}_{\mathsf{pk}} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{C} \times \mathcal{D}$ computes the commitment string c of fixed length and a decommitment value d from the message $m \in \mathcal{M}$. Very often $d = (m; r)$, where $r \in \mathcal{R}$ is the randomness used in the commitment. We also use a simplified version of the notation of the commitment function $\mathsf{Com}_{\mathsf{pk}} : \mathcal{M} \rightarrow \mathcal{C} \times \mathcal{D}$. Randomness is still used in the commitment process, even though it is not explicitly specified. The opening algorithm $\mathsf{Open}_{\mathsf{pk}} : \mathcal{C} \times \mathcal{D} \rightarrow \mathcal{M} \cup \{\perp\}$, given the correct commitment and decommitment values, outputs the message m. If the decommitment value is incorrect, the algorithm outputs the abort value $\perp$.

Definition 3.1. *A commitment scheme is (t, ε)-hiding, if any t-time adversary $A = (A_1, A_2)$ achieves advantage*

$$\mathrm{Adv}^{\mathsf{hid}}_{\mathsf{Com}}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}, s \leftarrow \{0,1\}, (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), \\ (c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) : A_2(\sigma, c) = s \end{array} \right] - 1 \right| \leqslant \varepsilon \ .$$

Two special cases of this property are statistical and perfect hiding. Statistical hiding is also known as ε-hiding, and means that if the adversary has infinite computing power, he gets information about the message being committed to with negligible, i.e., very, very small probability. Perfect hiding means that a commitment to a message reveals no information about the message, even to an infinitely powerful adversary. We use the term computationally hiding to refer to the cases where t is not infinite.

Definition 3.2. *A commitment scheme is (t, ε)-binding, if any t-time adversary A achieves advantage*

$$\mathrm{Adv}^{\mathsf{bind}}_{\mathsf{Com}}(A) = \Pr\left[\begin{array}{l}\mathsf{pk} \leftarrow \mathsf{Gen}, (c, d_0, d_1) \leftarrow A(\mathsf{pk}) : \\ \bot \neq \mathsf{Open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{Open}_{\mathsf{pk}}(c, d_1) \neq \bot\end{array}\right] \leqslant \varepsilon \ .$$

Two special cases of this property are statistical and perfect binding. Statistical binding is also known as ε-binding. This means that even if the adversary has infinite computing power, he can cheat with negligible probability. Perfect binding means that even with infinite computing power, the adversary cannot change his mind after committing to a message. We use the term computationally binding to refer to the cases where t is not infinite.

Theorem 3.1. *A commitment scheme cannot be statistically binding and statistically hiding at the same time.*

Proof. Without loss of generality we fix two messages m_0 and m_1. It suffices to show that a statistically binding commitment scheme cannot be statistically hiding. Let $A = (A_1, A_2)$ be an adversary that plays the hiding game. We let A_1 output the pair (m_0, m_1) and the challenger commits to one of the messages, outputting c. As the scheme is statistically binding, there exists a double opening for c with probability that is not greater than ε. This means that given enough time, A_2 can check whether c is a commitment of m_0 or m_1, thus, distinguishing between commitments of the two messages with probability $1 - \varepsilon$. □

This theorem implies that a commitment scheme cannot be perfectly binding and hiding at the same time either.

3.1 Simple Commitment Schemes

In this section we describe two simple commitment schemes that have no additional properties besides hiding and binding specified in Definition 3.1 and Definition 3.2. In that sense they are the simplest possible commitment schemes. In further sections we consider more common commitment schemes that have addditional properties.

3.1.1 Simplified Canetti-Fischlin Commitment Scheme

We describe a commitment scheme that is based on the weakest assumption that we discuss in this thesis—that pseudorandom generators exist. The Canetti-Fischlin commitment scheme [CF01] is quite complex as it achieves very strong security objectives. The simplified version we describe is similar to Naor's bit commitment

scheme [Nao89], and provides a nice example of bit commitment using pseudorandom generators. The construction uses a pseudorandom generator PRG that expands a bitstring of length k into a bitstring of length n for $k < n$.

Setup. The key generator algorithm produces a public key pk of length n by uniformly choosing a string from $\{0,1\}^n$.

$$\mathsf{pk} \leftarrow \mathsf{Gen} \ .$$

Commitment. The sender uniformly chooses a bitstring $d \in \{0,1\}^k$ that will also be used as the decommitment value. To commit to a bit $m \in \{0,1\}$, he does the following

$$\mathsf{Com}_{\mathsf{pk}}(m;r) = (c,d) = \begin{cases} (\mathsf{PRG}(d), d), & \text{if } m = 0, \\ (\mathsf{PRG}(d) \oplus \mathsf{pk}, d), & \text{if } m = 1. \end{cases}$$

Opening. To decommit, the the sender transfers d to the receiver who computes $\mathsf{PRG}(d)$ and checks whether $c = \mathsf{PRG}(d)$ or $c = \mathsf{PRG}(d) \oplus \mathsf{pk}$ to find out which value was committed to.

Theorem 3.2. *Let* PRG *be a* (t,ε)*-pseudorandom generator. Then the simplified Canetti-Fischlin commitment scheme is* $(\tau, 2\varepsilon)$-hiding *and* (2^{2k-n})-binding*, where* $\tau = t - \mathcal{O}(1)$.

Proof. Hiding. It is quite straightforward to see that if an adversary is able to distinguish $\mathsf{PRG}(d)$ and $\mathsf{pk} \oplus \mathsf{PRG}(d)$ then he is also able to tell the difference between $\mathsf{PRG}(d)$ and a bitstring chosen uniformly from $\{0,1\}^n$ because the value $\mathsf{pk} \oplus \mathsf{PRG}(d)$ has uniform distibution over that set. More formally, let A be an adversary that wins the hiding game $\mathcal{G}_{\mathsf{hid}}$ for this scheme

$\mathcal{G}^A_{\mathsf{hid}}$

$\mathsf{pk} \leftarrow \mathsf{Gen}$
$s \leftarrow \{0,1\}$
if $s = 0$, $c \leftarrow \mathsf{PRG}(d)$
else $c \leftarrow \mathsf{PRG}(d) \oplus \mathsf{pk}$
if $A(c) = s$, return 1
else, return 0

with advantage $\mathrm{Adv}(A)$. We change the game and substitute the value output by $\mathsf{PRG}(d)$ with a uniformly chosen bitstring $r \leftarrow \{0,1\}^n$. In this game $\widehat{\mathcal{G}}_{\mathsf{hid}}$

$\widehat{\mathcal{G}}^A_{\mathsf{hid}}$

$\lceil \mathsf{pk} \leftarrow \mathsf{Gen}$
$s \leftarrow \{0,1\}$
$r \leftarrow \{0,1\}^n$
if $s = 0$, $c \leftarrow r$
else $c \leftarrow r \oplus \mathsf{pk}$
if $A(c) = s$, return 1
$\lfloor$else, return 0

the distributions of $\mathsf{Com}_{\mathsf{pk}}(0)$ and $\mathsf{Com}_{\mathsf{pk}}(1)$ are both uniform. Hence, the probability that the adversary A guesses s correctly in the new game is $\frac{1}{2}$, because one uniform value could appear for commitment values of both of the messages, thus, making the guess of A independent from the message and rendering it wrong half of the time. This means that the advantage of the adversary in the game $\widehat{\mathcal{G}}^A_{\mathsf{hid}}$ is 0. Thus, the advantage $\mathrm{Adv}(A)$ must not be greater than 2ε, otherwise A is able to tell the difference between the value generated by $\mathsf{PRG}(d)$ and a uniformly chosen value. The adversary B can use the adversary A to win the pseudorandomness game

$B(c')$

$\lceil \mathsf{pk} \leftarrow \mathsf{Gen}$
$s \leftarrow \{0,1\}$
if $s = 0$, $c \leftarrow c'$
else $c \leftarrow c' \oplus \mathsf{pk}$
if $A(c') = s$, return 1
$\lfloor$else, return 0

The probability that A makes a correct guess if c' is a value from PRG is $\frac{1}{2} \pm \mathrm{Adv}(A)$, and the probability that A guesses right if c' is from uniform distribution is $\frac{1}{2}$ as established above. Thus, B wins the pseudorandomness game

$\mathcal{G}^B_{\mathsf{prg}}$

$$\left\lceil \begin{array}{l} s' \leftarrow \{0,1\} \\ d \leftarrow \{0,1\}^k \\ r \leftarrow \{0,1\}^n \\ \text{if } s' = 0,\ \ c' \leftarrow \mathsf{PRG}(d) \\ \text{if } s' = 1,\ \ c' \leftarrow r \\ \text{if } B(c') = s', \text{ return } 1 \\ \text{else, return } 0 \end{array} \right\rfloor$$

with probability

$$\begin{aligned} \Pr\left[B(c') = s'\right] =& \frac{1}{2}\Pr\left[B(c') = 1 | c' = r\right] + \frac{1}{2}\Pr\left[B(c') = 0 | c' = \mathsf{PRG}(d)\right] \\ =& \frac{1}{2}\Pr\left[A(c') = s | c' = r\right] + \frac{1}{2}(1 - \Pr\left[A(c') = s | c' = \mathsf{PRG}(d)\right]) \\ =& \frac{1}{2} \pm \frac{1}{2}\operatorname{Adv}(A) \end{aligned}$$

but this contradicts the assumption that PRG is a (t, ε)-pseudorandom generator if $\operatorname{Adv}(A) > 2\varepsilon$, and the claim follows.

Binding. To win the binding game, the adversary A has to output a commitment c and a valid double opening d_0, d_1 for c. It is quite easy to note that, in this case, $\mathsf{PRG}(d_0) = \mathsf{PRG}(d_1) \oplus \mathsf{pk}$ which in turn means that $\mathsf{pk} = \mathsf{PRG}(d_0) \oplus \mathsf{PRG}(d_1)$. We will call these kinds of keys insecure public keys and denote their group by $\widehat{K}$. There are 2^{2k} possibilities to choose a pair (d_0, d_1) and, thus, there are less than 2^{2k} values of insecure public keys. As there are 2^n possibilities to choose a public key, the probability that $\mathsf{pk} \in \widehat{K}$ and that a double opening even exists, is less than $2^{2k}/2^n = 2^{2k-n}$. □

It is interesting to note that in this scheme a public key pk can be used in commitments more than once. Canetti and Fischlin require that in their scheme pk be used only once due to the stricter security requirements they describe.

As an instantiation, AES [Nat01] in counter mode can be used as the pseudorandom generator in the scheme. For an ε-binding commitment scheme, safe values for n are greater than $2k - \log_2 \varepsilon$, because

$$\begin{aligned} \varepsilon &\geq 2^{2k-n} \\ \log_2 \varepsilon &\geq 2k - n \\ n &\geq 2k - \log_2 \varepsilon \ . \end{aligned}$$

3.1.2 Collision-Resistant Hash Function Commitment

A commitment from collision-resistant hash functions was proposed by Shai Halevi and Silvio Micali [HM96]. Let n be the length of the message being committed to and let k be the length of the security parameter. Let $\mathcal{H}$ be a family of collision resistant hash functions and $\mathcal{F}$ be a universal family of hash functions from $\{0,1\}^L$ to $\{0,1\}^n$, where $L = 4k + 2n + 4$. A hash function from the universal function family $\mathcal{F}_{m,r} = \{f : f(r) = m\}$ can be easily sampled—for a fixed $m \in \{0,1\}^n$ and $r \in \{0,1\}^L$, it is easy to find functions f such that $f(r) = m$ and uniformly choose one of them.

Setup. The generation algorithm takes a random collision-resistant function $h : \{0,1\}^L \to \{0,1\}^k$ from $\mathcal{H}$ and outputs the public key:

$$h \leftarrow \mathsf{Gen} \ .$$

Commitment. To commit to a message $m \in \{0,1\}^n$, the sender uniformly picks a value $r \leftarrow \{0,1\}^L$ and computes $y = h(r)$. Next, the sender uniformly picks a function $f \leftarrow \mathcal{F}_{m,r}$. The commitment function outputs a commitment-decommitment pair

$$\mathsf{Com}_{\mathsf{pk}}(m; r) = ((f, y), (m; r)) \ .$$

Opening. To decommit, the sender sends m and r to the receiver. The receiver now checks whether $y = h(r)$ and computes $m = f(r)$.

Theorem 3.3. *Let $\mathcal{H}$ be a (t, ε)-collision resistant hash function family. Then the Halevi-Micali commitment scheme is* statistically hiding *and* (τ, ε)-binding*, where* $\tau = t - \mathcal{O}(1)$.

Proof. Hiding. The proof of the statistical hiding property of this scheme is very technical and we do not summarise it here. A detailed proof can be found on pages 211–213 in the article [HM96]. Most of the steps in the proof are quite straightforward, except for the double counting argument that uses properties of universal hash function families, especially the pairwise independence property.

Binding. It is quite straightforward to see that if an adversary is able to create a double opening for a commitment then he is also able to find a collision in the hash function h that is assumed to be collision resistant.

More formally, let B be an adversary that wins the binding game with advantage $\mathrm{Adv}(B)$. Let the commitment be (f, y) and the double opening be $d_0 = (m_0; r_0)$ and $d_1 = (m_1; r_1)$. If these decommitments are both valid then $h(r_0) =$

$h(r_1)$ holds and, thus, the adversary B can find a collision for the hash function h with advantage $\mathrm{Adv}(B)$. Since we assume that h is collision resistant, $\mathrm{Adv}(B) \leq \varepsilon$. □

As an instantiation SHA-256 can be used as the hash function h from the collision resistant family and f can be chosen from the universal hash function family:

$$\mathcal{F} = \left\{ f_{A,b} : A \in \{0,1\}^{nt}, b \in \{0,1\}^{n} \right\} \quad ,$$

where $f_{A,b}(x)$ is defined as $f_{A,b}(x) = Ax + b$.

4 Duality between Encryption and Commitment

When one looks at commitment and encryption schemes, it is rather easy to spot the similarities between the two cryptographic primitives. Although their functionalities are different and they are used for different purposes, they share a similar structure. Both encryptions and commitments make use of public keys, encryption schemes have an additional secret key. In both cases the secret hiding and revealing stages occur, only the way these phases are handled differ. In the case of an encryption scheme, we use the public key to lock a message and the secret key to open it again. Commitment schemes also use the public key to lock a message, but to instead of unlocking, one usually needs to reveal the message in order to open a commitment.

It is always possible to make a commitment scheme from an encryption scheme. Making an encryption scheme from a commitment scheme is trickier though, because there exists no secret key in the simple commitment scheme construction. In order to make an encryption scheme from a commitment scheme, the latter needs to be extractable. The trapdoor information that an extractable commitment can reveal is not used in commitment schemes, because it would break their security. However, extractability often simplifies security proofs of complex two and multiparty computations such as the cointossing protocol. The idea is to use the original commitment scheme in the protocol and the modified version in the security proof, thus making the proofs simpler and allowing the contents of the commitment to be revealed using the trapdoor.

4.1 Encryption Schemes

First of all, we will describe encryption schemes in general. The structure of a commitment scheme, described in Chapter 3, is very similar to that of an encryption scheme. The encryption scheme also consists of three functions: key generation Gen, encryption Enc and decryption Dec. The key generation algorithm generates the public and secret key $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$, revealing only the public key to all parties. The encryption algorithm $\mathsf{Enc}_{\mathsf{pk}} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{E}$ computes the cryptotext e of fixed length from the message $m \in \mathcal{M}$ and the randomness $r \in \mathcal{R}$. Similarly to the commitment scheme case, we also give the formalisation of the randomised encryption function without the specified randomness $\mathsf{Enc}_{\mathsf{pk}} : \mathcal{M} \rightarrow \mathcal{E}$. The decryption algorithm $\mathsf{Dec}_{\mathsf{sk}} : \mathcal{E} \rightarrow \mathcal{M} \cup \{\perp\}$, given the encryption value, outputs the message m. If the encryption value is corrupt, the algorithm outputs the abort value $\perp$.

We define two properties of encryption schemes. An encryption scheme can be *indistinguishable under chosen plaintext attack* (IND-CPA). This property defines the security of the encryption scheme against a time-bounded adversary, who

outputs two messages and given the encryption of one of the two, has to decide which one has been encrypted. We give a more formal definition of the IND-CPA property [GM82].

Definition 4.1. *An encryption scheme is (t, ε)-IND-CPA secure, if a t-time adversary $A = (A_1, A_2)$ achieves advantage*

$$\mathrm{Adv}^{\mathsf{ind-cpa}}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), \\ e \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_s) : A_2(\sigma, e) = s \end{array} \right] - 1 \right| \leqslant \varepsilon \ .$$

It is easy to see the similarities between this definition and Definition 3.1 of the hiding property of commitment schemes.

An encryption schemes can be *indistinguishable under adaptive chosen ciphertext attack* (IND-CCA2). This property defines the security of the encryption scheme against a time-bounded adversary A that works much like the adversary in Definition 4.1, but in addition it has access to the decryption oracle at two stages of the game—first, when outputting the message pair, and second, when trying to determine which of the two messages was encrypted. It is assumed that A does not ask the oracle to decrypt e. We give a more formal definition of the IND-CCA2 property [RS91].

Definition 4.2. *An encryption scheme is (t, ε)-IND-CCA2 secure, if any t-time adversary $A = (A_1, A_2)$ achieves advantage*

$$\mathrm{Adv}^{\mathsf{ind-cca2}}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma) \leftarrow A_1^{\mathsf{Dec}_{\mathsf{sk}}(\cdot)}(\mathsf{pk}), \\ e \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_s) : A_2^{\mathsf{Dec}_{\mathsf{sk}}(\cdot)}(\sigma, e) = s \end{array} \right] - 1 \right| \leqslant \varepsilon \ ,$$

where $\mathsf{Dec}_{\mathsf{sk}}(\cdot)$ is a decryption oracle and A_2 cannot make the oracle query $\mathsf{Dec}_{\mathsf{sk}}(e)$ to have e decrypted.

4.2 Extractability

The notion of extractable commitments was proposed in the article [SCP00] in the context of non-interactive zero-knowledge proofs. Extractable commitment schemes have an additional property to the usual hiding and binding—if a party knows a certain secret value, they are able to extract the message from the commitment. There is an extra key generation algorithm Gen^* in an extractable commitment scheme. This algorithm outputs the secret key sk in addition to the public key pk. Everything else in the scheme works as before only there exists an

additional function. The extraction function $\mathsf{Extr}_{\mathsf{sk}} : \mathcal{C} \to \mathcal{M}$ opens a commitment $c \in \mathcal{C}$ to the original message $m \in \mathcal{M}$. We give the formal definition for extractability [SCP00, Cre02, LAN05].

Definition 4.3. *A commitment scheme is* (t, ε)-extractable *if any* t*-time adversary* A *achieves advantage*

$$\mathrm{Adv}^{\mathsf{extr}}(A) = \Pr\left[\begin{array}{l}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}^*, (c, d) \leftarrow A(\mathsf{pk}) : \\ \mathsf{Extr}_{\mathsf{sk}}(c) \neq \mathsf{Open}_{\mathsf{pk}}(c, d) \neq \perp\end{array}\right] \leqslant \varepsilon \ , \tag{4}$$

where the distributions of the public keys pk *output by* Gen *and* Gen^* *coincide.*

Less formally, there is only a negligible chance that a time-bounded adversary A can create such a commitment-decommitment pair (c, d) that the function $\mathsf{Extr}_{\mathsf{sk}}$ extracts a message m from the commitment, while $\mathsf{Open}_{\mathsf{pk}}(c, d)$ outputs a different message m'. The original Gen function is used, when we want to initiate the commitment scheme. The second generation function Gen^* is used, when we want to initiate the scheme that is equivalent to an encryption scheme. The extraction function can only be used in the second case. It is quite simple to see that as soon as the receiver knows sk, the commitment scheme is useless, because the receiver can open it at any time.

Theorem 4.1. *For every reasonable time bound* t*, a* (t, ε)*-extractable commitment scheme is only computationally hiding.*

Proof. Let $\mathcal{C}om = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Com}, \mathsf{Open}, \mathsf{Extr})$ be an extractable commitment scheme then for any t-time adversary A the inequality (4) holds. Let us fix the adversary as the challenger in the hiding game. For some fixed and valid messages $m_0, m_1 \in \mathcal{M}$, the adversary A will be the following

$A(\mathsf{pk})$

$$\left[\begin{array}{l}s \leftarrow \{0, 1\} \\ (c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) \\ \mathsf{Return}\ (c, d)\end{array}\right.$$

From the extractability condition and the commitment c that A outputs,

$$\Pr\left[\mathsf{Extr}_{\mathsf{sk}}(c) \neq \mathsf{Open}(c, d) = m_i\right] \leq \varepsilon \ . \tag{5}$$

Now consider an adversary $B = (B_1, B_2)$ who plays the hiding game, finding the secret key and, thus, computing $\mathsf{Extr}_{\mathsf{sk}}(c)$. That adversary achieves advantage

$$\mathrm{Adv}(B) = \Pr\left[\begin{array}{l}\mathsf{pk} \leftarrow \mathsf{Gen}, s \leftarrow \{0, 1\}, (m_0, m_1, \sigma) \leftarrow B_1(\mathsf{pk}), \\ c \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) : B_2(\sigma, m_s) = s\end{array}\right] > 1 - \varepsilon \ .$$

The only problem that remains is how the adversary B can find the secret key. First, we look at schemes where for any fixed pk there exists exactly one sk. In this case, B simply runs the Gen^* function until he finds the secret key corresponding to the given public key pk. This means that the scheme cannot be statistically hiding, because given enough time, the adversary finds the message with a very high probability.

Secondly, we look at schemes, where for each pk there exist one or more sk. Then for a fixed pk it is possible to choose sk^* that achieves the best error probability against A. This probability can be explicitly computed by generating all possible keys, computing the commitments for m_0 and m_1, and finding the corresponding probabilities. Obviously the inequality (5) must still hold since sk^* can be taken as sk. Hence, the best error probability is not greater than ε. For the same reasons, these schemes are also only computationally hiding. □

Theorem 4.1 implies that the extraction function must be efficient. Since the scheme is computationally hiding, it can be opened if we have enough time, and we do not need the secret key at all. So we do not require a separate inefficient extraction function that needs a secret key. Intuitively we can say that the scheme must be at least statistically binding. Otherwise there exist double openings for at least some commitments and the extraction function could not uniquely open the commitments. Unfortunately, this is not always true [Cre02].

4.3 Canonical Correspondence

We show how to construct an encryption scheme from a commitment scheme and *vice versa*. In the following, let $\mathcal{C}om = (\mathsf{Gen}_{\mathcal{C}om}, \mathsf{Gen}^*_{\mathcal{C}om}, \mathsf{Com}, \mathsf{Open}, \mathsf{Extr})$ be an extractable commitment scheme and $\mathcal{E}nc = (\mathsf{Gen}_{\mathcal{E}nc}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme. Also, let $m \in \mathcal{M}$ be a message and $r \in \mathcal{R}$ be the used randomness.

First, to construct an encryption scheme from an extractable commitment scheme, we map the functions of $\mathcal{C}om$ to those of $\mathcal{E}nc$. The encryption scheme needs a key generation, an encryption and a decryption function.

We use the key generation algorithm $\mathsf{Gen}^*_{\mathcal{C}om}$ of the commitment scheme as the key generation algorithm $\mathsf{Gen}_{\mathcal{E}nc}$ of the encryption scheme. The $\mathsf{Gen}_{\mathcal{E}nc}$ function outputs the public and secret key pair $(\mathsf{pk}, \mathsf{sk})$ that it gets from $\mathsf{Gen}^*_{\mathcal{C}om}$. To show canonical correspondence between a commitment function and an encryption function, we also need to specify the randomness r used in both schemes. Therefore, we use the functions with specified randomness. The $\mathsf{Enc}(m; r)$ function uses the $\mathsf{Com}_{\mathsf{pk}}(m; r)$ function to encrypt the message m, outputting the commitment part c from the commitment-decommitment pair (c, d) that it gets from $\mathsf{Com}_{\mathsf{pk}}(m; r)$. The decryption function $\mathsf{Dec}(c)$ uses the extraction function $\mathsf{Extr}_{\mathsf{sk}}(c)$ of the commitment scheme to open the encryption and output the message m. It is straight-

forward from the Definition 4.3 that the decryption function succeeds with very probability $(1 - \varepsilon)$.

Next, to construct a commitment scheme from an encryption scheme, we map the functions of $\mathcal{E}nc$ to those of $\mathcal{C}om$. An ordinary commitment scheme needs a key generation, a commitment and an opening function. We also add an extraction function and a second key generation algorithm that outputs the key pair. This way the constructed commitment scheme is extractable.

The $\mathsf{Gen}_{\mathcal{E}nc}$ function outputs a key pair $(\mathsf{pk}, \mathsf{sk})$. We only need the public key for the original key generation algorithm $\mathsf{Gen}_{\mathcal{C}om}$ of the commitment scheme. This algorithm takes pk from the pair and discards the rest, as in the commitment scheme scenario, the secret key is not known to anyone. The additional key generation function $\mathsf{Gen}^*_{\mathcal{C}om}$, on the other hand, outputs the whole key pair $(\mathsf{pk}, \mathsf{sk})$ that it received from $\mathsf{Gen}_{\mathcal{E}nc}$. The commitment function $\mathsf{Com}_{\mathsf{pk}}(m; r)$ outputs the commitment-decommitent pair (c, d), where c is the encryption of the message m output by $\mathsf{Enc}(m; r)$, and d simply contains the message m and the used randomness r. The opening function $\mathsf{Open}(c, d)$ recommits to the message m with randomness r and outputs the message m if the commitment it computed matches c; otherwise, the function outputs a special character $\perp$. The extraction function has to open the commitment c without the decommitment value. The function $\mathsf{Extr}_{\mathsf{sk}}(c)$ outputs the message m output by $\mathsf{Dec}(c)$.

The described transformations provide a canonic correspondence between encryption schemes and commitment schemes.

4.4 IND-CPA Security and Extractability

We will show the duality between the IND-CPA security property of encryption schemes and the computational hiding property of commitment schemes. In the following, let $\mathcal{E}nc = (\mathsf{Gen}_{\mathcal{E}nc}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and $\mathcal{C}om = (\mathsf{Gen}_{\mathcal{C}om}, \mathsf{Gen}^*_{\mathcal{C}om}, \mathsf{Com}, \mathsf{Open}, \mathsf{Extr})$ be an extractable commitment scheme.

Theorem 4.2. *Let $\mathcal{C}om$ and $\mathcal{E}nc$ be in canonical correspondence. Then (t, ε)-hiding implies (t, ε)-IND-CPA security.*

Proof. We show that the constructed encryption scheme $\mathcal{E}nc$ has the necessary properties, i.e., it is IND-CPA secure. We show that any time-bounded adversary that complies to the assumption that $\mathcal{C}om$ is computationally hiding, is also subject to the IND-CPA security property. To do this, we take an adversary A that plays the hiding game and transform it into an adversary that plays the IND-CPA game. A time-bounded $A = (A_1, A_2)$, playing the hiding game, achieves advantage

$$\mathrm{Adv}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}_{\mathcal{C}om}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), \\ (c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) : A_2(\sigma, c) = s \end{array} \right] - 1 \right| \leq \varepsilon \ .$$

Instead of $\mathsf{Gen}_{\mathcal{C}om}$ used above, the challenger can use $\mathsf{Gen}^*_{\mathcal{C}om}$, since the distributions of public keys output by both algorithms coincide, and just discard the generated secret key from the acquired pair $(\mathsf{pk}, \mathsf{sk})$. In that case, and considering that $\mathcal{C}om$ and $\mathcal{E}nc$ are in canonical correspondence, we can rewrite the advantage as

$$\mathrm{Adv}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathcal{E}nc}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), \\ c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_s) : A_2(\sigma, c) = s \end{array} \right] - 1 \right| \leq \varepsilon \ .$$

Hence the corresponding encryption scheme $\mathcal{E}nc$ is (t, ε)-IND-CPA secure. □

Theorem 4.3. *Let $\mathcal{E}nc$ and $\mathcal{C}om$ be in canonical correspondence. Then (t, ε)-IND-CPA security implies (t, ε)- hiding.*

Proof. We show that the constructed commitment scheme $\mathcal{C}om$ has the necessary properties, i.e., it is computationally hiding. We show that any time-bounded adversary that adheres to the assumption that $\mathcal{E}nc$ is IND-CPA secure, is also subject to the computational hiding property. We do this similarly to the proof of the previous theorem—we take an adversary A that plays the IND-CPA game and transform it into an adversary that plays the hiding game. A time-bounded adversary $A = (A_1, A_2)$, playing the IND-CPA game, achieves advantage

$$\mathrm{Adv}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}_{\mathcal{E}nc}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), e \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_s) : \\ A_2(\sigma, e) = s \end{array} \right] - 1 \right| \leq \varepsilon \ .$$

Considering the canonical correspondence between $\mathcal{E}nc$ and $\mathcal{C}om$, and as the distributions of the public keys output by $\mathsf{Gen}_{\mathcal{C}om}$ and $\mathsf{Gen}^*_{\mathcal{C}om}$ coincide, we can rewrite the advantage as

$$\mathrm{Adv}(A) = \left| 2 \cdot \Pr \left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}_{\mathcal{C}om}, s \leftarrow \{0,1\}, (m_0, m_1, \sigma) \leftarrow A_1(\mathsf{pk}), \\ (e, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) : A_2(\sigma, e) = s \end{array} \right] - 1 \right| \leq \varepsilon \ .$$

Hence the corresponding commitment scheme $\mathcal{C}om$ is (t, ε)-hiding. □

4.5 An Example of Canonical Correspondence

As an example to canonical correspondence between encryption and commitment schemes, we talk about the correspondence between the ElGamal encryption [Gam84] and commitment scheme. The ElGamal scheme is based on the DDH assumption and is, therefore, more difficult to instantiate as the DDH assumption is stronger than the discrete logarithm assumption. The instantiation is usually for certain subgroups of $\mathbb{Z}_p$, where p is prime and groups defined over certain elliptic curves. In the following, we assume that $\mathbb{G}_n$ is a public parameter.

4.5.1 ElGamal Encryption Scheme

Setup. A generator g is chosen from $\mathbb{G}_n$. A value $x \in \mathbb{Z}_n$ is chosen and the generation algorithm Gen produces a public key:

$$\mathsf{pk} = (g, g^x) \ ,$$

where x is the secret key.

Encryption. To encrypt a message $m \in \mathcal{M}$, it is first converted into an element of $\mathbb{G}_n$, the value r is uniformly chosen from $\mathbb{Z}_n$ and the encryption is computed in the following way

$$\mathsf{Enc}_{\mathsf{pk}}(m; r) = (g^r, mg^{xr}) \ .$$

Decryption. To decrypt the ciphertext (g^r, mg^{xr}), the secret key x is used to compute

$$\mathsf{Dec}_{\mathsf{sk}}(g^r, mg^{xr}) = \frac{mg^{xr}}{(g^r)^x} = m \ .$$

4.5.2 ElGamal Commitment Scheme

We derive the key generation algorithm from that of the encryption scheme using the canonical correspondence. It is modified so that it only outputs the public key from the original key pair.

Setup. A generator g is chosen from $\mathbb{G}_n$. The generation algorithm Gen produces a public key:

$$\mathsf{pk} = (g, g^x) \ ,$$

where no-one knows the value of x, i.e., the secret key sk is deleted.

Commitment. To commit to the message $m \in \mathbb{Z}_n$, the sender chooses a random $r \in \mathbb{Z}_n$ and computes

$$\mathsf{Com}_{\mathsf{pk}}(m; r) = ((g^r, g^m g^{xr}), (m; r)) \ .$$

Opening. To open the commitment, the sender sends m and r. The receiver can compute the pair $(g^r, g^m g^{xr})$ to check whether the commitment is valid.

There is a slight difference between the classical form of the ElGamal encryption function and the commitment function. The function $\mathsf{Enc}(m; r)$ is usually given as (g^r, mg^{xr}), whereas the commitment function outputs $c = g^r, g^m g^{xr}$. We note that the form used in the commitment can also be used as an encryption function. In that case, there is no need to first convert the message into an element of $\mathbb{G}_n$, as it can simply be a value from $\mathbb{Z}_n$. To find the value of m on decryption, g^m is computed as m was before and an algorithm for finding the discrete logarithm can be applied to find the value of m. This is useful, if the values of m are small and, for example, Pollard's lambda algorithm or a lookup table can be used. This form of the ElGamal encryption is known as lifted ElGamal and it has an additional property that can be very useful, namely, it is an additively homomorphic encryption scheme. We discuss homomorphic schemes in Sec. 6.1.

Theorem 4.4. *Let $\mathbb{G}_n$ be (t, ε)-DDH group. Then the ElGamal commitment scheme is* $(\tau, 2\varepsilon)$-hiding *and* perfectly binding.

Proof. HIDING. To prove that the scheme is (τ, ε) hiding, we show how to convert any τ-time adversary $A = (A_1, A_2)$ against the hiding property to an adversary B that can solve the decisional Diffie-Hellman problem. Assume that A wins the hiding game $\mathcal{G}_{\mathsf{hid}}$ for this scheme

$\mathcal{G}^A_{\mathsf{hid}}$

$(g, g^x) \leftarrow \mathsf{Gen}$
$(m_0, m_1, \sigma) \leftarrow A_1(g, g^x)$
$s \leftarrow \{0, 1\}$
$r \leftarrow \mathbb{Z}_n$
$c \leftarrow (g^r, g^{m_s} g^{xr})$
if $A_2(\sigma, c) = s$, return 1
else, return 0

with advantage $\mathrm{Adv}(A)$. We change the game and substitute the value of g^{xr} with a uniformly chosen element z from $\mathbb{G}_n$. In this game $\widehat{\mathcal{G}}_{\mathsf{hid}}$

$\widehat{\mathcal{G}}^A_{\mathsf{hid}}$

$(g, g^x) \leftarrow \mathsf{Gen}$
$(m_0, m_1, \sigma) \leftarrow A_1(g, g^x)$
$s \leftarrow \{0, 1\}$
$r \leftarrow \mathbb{Z}_n$
$z \leftarrow \mathbb{G}_n$
$c \leftarrow (g^r, g^{m_s} z)$
if $A_2(\sigma, c) = s$, return 1
else, return 0

the probability that the adversary A guesses s correctly is $\frac{1}{2}$ because the guess of A is independent from the message m_s that was committed to. To be more explicit, c is totally independent from s and, thus, the output of A is independent from s. This means that the advantage of the adversary in the game $\widehat{\mathcal{G}}_{\mathsf{hid}}$ is 0. Thus, the advantage $\mathrm{Adv}(A)$ must not be greater than 2ε, otherwise A is able to tell the difference between a DDH-tuple and a tuple with a uniformly chosen element. The adversary B can use the adversary A to win the DDH game

$B(g, g^a, g^b, y)$

$(m_0, m_1, \sigma) \leftarrow A_1(g, g^a)$
$s \leftarrow \{0, 1\}$
$c \leftarrow (g^b, g^{m_s} y)$
if $A_2(\sigma, c) = s$, return 1
else, return 0

Observe that the working time of B is $t = \tau + \mathcal{O}(1)$. The probability that A guesses right if $y = g^{ab}$ is $\frac{1}{2} \pm \mathrm{Adv}(A)$, and if y is a uniformly chosen value from $\mathbb{G}_n$, the probability is $\frac{1}{2}$ as established above. Thus, B wins the DDH game

$\mathcal{G}^B_{\mathsf{DDH}}$

$g \leftarrow \mathsf{Gen}$
$s' \leftarrow \{0, 1\}$
$a, b \leftarrow \mathbb{Z}_n$
$y' \leftarrow \mathbb{G}_n$
if $s' = 0$, $y \leftarrow g^{ab}$
if $s' = 1$, $y \leftarrow y'$
if $B(g, g^a, g^b, y) \leftarrow s'$, return 1
else, return 0

with probability

$$\begin{aligned}\Pr\left[B(g,g^a,g^b,y)=s'\right] =& \frac{1}{2}\Pr\left[B(g,g^a,g^b,y)=1|y=y'\right] \\ &+\frac{1}{2}\Pr\left[B(g,g^a,g^b,y)=0|y=g^{ab}\right] \\ =& \frac{1}{2}\Pr\left[A_2((g^b,g^{m_s}y))=s|y=y'\right] \\ &+\frac{1}{2}(1-\Pr\left[A((g^b,g^{m_s}y))=s|y=g^{ab}\right]) \\ =& \frac{1}{2}\pm\frac{1}{2}\operatorname{Adv}(A)\end{aligned}$$

but this contradicts the assumption that $\mathbb{G}_n$ is a (t,ε)-DDH group, if $\operatorname{Adv}(A)>2\varepsilon$.

Binding. Next we prove that the scheme is perfectly binding. Let us assume that $c=(g^r,g^m g^{xr})$ is the commitment of $(m;r)$. As g is part of the public key, then given enough time, it is possible to find the secret key x of the corresponding encryption scheme, as $x=\log g^x$. It is straightforward to see that as soon as we know this key, we are able to decrypt the commitment, as if it was an encryption, by computing $\frac{g^m g^{xr}}{(g^r)^x}$, and find the uniquely determined value of $\log g^m=m$. □

5 Simulatable Commitments

5.1 Equivocability

Equivocable commitment schemes, as defined in [CIO98, DG03, LAN05], add two functions—$\mathsf{Com}^*_{\mathsf{sk}}$ and Equiv—to the original commitment scheme construction and they also use the Gen^* algorithm like extractable schemes. We require that the distributions of the public keys output by Gen and Gen^* coincide. Alternatively, the case, where the two functions do not necessarily output keys with the same distribution is discussed in the article [DG03]. The assumption we make lets us write down the definition in a way that is easier to understand and this way equivocability works for every public key output by Gen.

The function $\mathsf{Com}^*_{\mathsf{sk}} : \emptyset \to \mathcal{C} \times \mathcal{S}$ outputs a fake commitment $c \in \mathcal{C}$ and additional information $\sigma \in \mathcal{S}$. The commitment c can be opened to any message $m \in \mathcal{M}$ with the help of the function $\mathsf{Equiv}_{\mathsf{sk}} : \mathcal{M} \times \mathcal{C} \times \mathcal{S} \to \mathcal{D}$ that outputs a decommitment value $d \in \mathcal{D}$ for the fake commitment $c \in \mathcal{C}$ and a chosen message $m \in \mathcal{M}$ so that $\mathsf{Open}(c, \mathsf{Equiv}_{\mathsf{sk}}(m, c, \sigma)) = m$.

Definition 5.1. *A commitment scheme is* (t, ε)-equivocable*, if any* t *time adversary* A *achieves advantage*

$$\mathrm{Adv}^{\mathsf{equiv}}(A) = \left| \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}^*, s \leftarrow \{0,1\} : \\ A^{\mathcal{O}(\cdot)}(\mathsf{pk}) = s \end{array} \right] - \frac{1}{2} \right| \leq \varepsilon \ .$$

The distributions of the public keys output by Gen *and* Gen^* *coincide. The oracle* $\mathcal{O}(\cdot)$ *does the following for a given message* m*:*

- *if* $s = 0$*, then* $\mathcal{O}(\cdot)$ *outputs* $(c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m; r)$*,*
- *if* $s = 1$*, then* $\mathcal{O}(\cdot)$ *computes* $(c, \sigma) \leftarrow \mathsf{Com}^*_{\mathsf{sk}}$ *and outputs* $(c, \mathsf{Equiv}_{\mathsf{sk}}(m, c, \sigma))$*.*

Less formally, there is only a negligible chance that a time-bounded adversary A can distinguish between real and fake commitment-decommitment pairs queried adaptively. Similarly to extractability, the original Gen function is used, when we want to initiate the commitment scheme and the second key generation function Gen^* is used, when we want to generate a fake commitment. It is possible to interchange these two key generation functions as the distributions of the public keys output by Gen and Gen^* coincide. It is quite simple to see that as soon as the sender knows sk, the commitment scheme is useless, because the sender can generate any decommitment value from a fake commitment.

Similarly to extractable commitment schemes, the trapdoor information that an equivocable commitment can reveal is not used in commitment schemes because it would break their security. However, equivocability often simplifies security

proofs, so the original commitment scheme is used in the protocol and the modified version can be used in the security proof. In the public random string model an equivocable commitment scheme can be constructed from any given commitment scheme [CIO98].

Theorem 5.1. *For every reasonable time bound t, a (t, ε)-equivocable commitment scheme is only computationally binding.*

Proof. Let $\mathcal{C}om = (\mathsf{Gen}, \mathsf{Gen}^*, \mathsf{Com}, \mathsf{Com}^*, \mathsf{Open}, \mathsf{Equiv})$ be an equivocable commitment scheme. Consider an adversary A who plays the binding game. The adversary is given the commitment value c and it needs to output two decommitment values d_0 and d_1 so that they open to different values $\perp \neq \mathsf{Open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{Open}_{\mathsf{pk}}(c, d_1) \neq \perp$. If the adversary knows the secret key, he can compute these decommitment values using the equivocability function $\mathsf{Equiv}_{\mathsf{sk}}$. The equivocability function works with probability 1, so the probability of A succeeding is also 1, if he knows the corresponding secret key.

It remains to show how the adversary A can find the secret key. This is easier than in the proof of Theorem 4.1. For every secret key output by Gen^*, the adversary tries whether it is possible to construct a double opening for the given commitment with the secret key sk. We assumed that the functions Gen and Gen^* give output with the same distribution, hence for every public key there exists at least one suitable secret key. Given enough time, the adversary is thus able to construct a double opening, making the commitment scheme at most computationally binding. □

5.2 Commitments Based on Discrete Logarithm

In this subsection we give two examples of commitment schemes based on the assumption that the discrete logarithm problem is hard: the Pedersen commitment scheme and the Fujisaki-Okamoto commitment scheme. The latter is also based on the assumption that factoring is hard.

5.2.1 Pedersen Commitment Scheme

The first discrete logarithm commitment scheme we discuss was proposed by Torben Pryds Pedersen [Ped91]. In the article, Pedersen defines the scheme in a subgroup of $\mathbb{Z}_p^*$, where the discrete logarithm problem is hard. But in principle, any group where the discrete logarithm problem is hard, can be used, for example DL-groups defined by elliptic curves. We give an abstraction of this scheme to an arbitrary DL-group $\mathbb{G}_q = \langle g \rangle$ that is publicly known and where q is a prime. All of the logarithms in this scheme are taken with respect to base g.

Setup. Two elements g and y of $\mathbb{G}_q$ are chosen such that $y \neq 1$ and nobody knows $\log y$. The algorithm Gen outputs the public key

$$\mathsf{pk} = (g, y) \ .$$

Commitment. To commit to the message $m \in \mathbb{Z}_q$, the sender chooses a random $r \in \mathbb{Z}_q$ and computes

$$\mathsf{Com}_{\mathsf{pk}}(m; r) = (g^m y^r, (m; r)) \ .$$

Opening. The commitment can only be opened by revealing m and r. The receiver can then check whether the commitment was indeed made to the value that the sender claims it to be.

Theorem 5.2. *Let $\mathbb{G}_q$ be (t, ε)-DL group. Then the Pedersen commitment scheme is* perfectly hiding *and* (τ, ε)-binding*, where $\tau = t - \mathcal{O}(1)$.*

Proof. Hiding. First, we show that the scheme is perfectly hiding. This means that $\mathsf{Com}_{\mathsf{pk}}(m; r)$ reveals no information about m. This holds, because r is chosen uniformly from $\mathbb{Z}_q$, and, thus $g^m y^r$ also has uniform distribution over $\mathbb{G}_q$, independently of the choice of m.

Binding. To prove that the scheme is computationally binding we show how to convert a τ-time adversary A against the binding property to an adversary B that can solve the discrete logarithm problem. Assume that A achieves advantage

$$\mathrm{Adv}^{\mathsf{bind}}(A) = \Pr\left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}, (c, d_0, d_1) \leftarrow A(\mathsf{pk}) : \\ \bot \neq \mathsf{Open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{Open}_{\mathsf{pk}}(c, d_1) \neq \bot \end{array}\right] \leq \varepsilon \ .$$

Our task is now to find an adversary B that uses A to find the discrete logarithm $x \in \mathbb{Z}_q$ of an element. If A outputs a valid double opening then $\mathsf{Com}_{\mathsf{pk}}(m_0; r_0) = \mathsf{Com}_{\mathsf{pk}}(m_1; r_1)$ and, thus, $g^{m_0} y^{r_0} = g^{m_1} y^{r_1}$. We note that as $m_0 \neq m_1$ then $r_0 \neq r_1$ and, hence, B can use the collision extraction property of discrete logarithm from Lemma 2.2 to find $x = \log y$. Consider an adversary B that gets the public key (g, y) as input and forwards the key to the adversary A, who has to output two message-random pairs, so that the commitments have the same value

$$\begin{array}{l} B(g, y) \\ \left[\begin{array}{l} (c, (m_0; r_0), (m_1; r_1)) \leftarrow A(g, y) \\ \mathsf{Return}\ x \leftarrow \dfrac{m_0 - m_1}{r_1 - r_0} \end{array}\right. \end{array}$$

The output of B is always the discrete logarithm of y provided that A outputs a valid double opening. Hence, $\mathrm{Adv}(B) = \mathrm{Adv}(A)$. To conclude the proof, observe that the working time of B is $t = \tau + \mathcal{O}(1)$. □

Instantiaton. In the original instantiation proposed by Pedersen, p and q are large primes such that q divides $p-1$, $\mathbb{G}_q$ is the unique subgroup of $\mathbb{Z}_p^*$ of order q, and g is a generator of $\mathbb{G}_q$. As any element $g \neq 1$ in $\mathbb{G}_q$ generates the group, the discrete logarithm with respect to base g is defined.

5.2.2 Fujisaki–Okamoto Commitment Scheme

The Fujisaki-Okamoto commitment scheme [FO97] is an extension of the Pedersen commitment scheme to the RSA [RSA78] modulus. Let $N = PQ$ be chosen from $\mathsf{RSA\text{-}mod}(w)$. Hence, P and Q are two large safe primes and there exist odd primes p and q such that $p = (P-1)/2$, $q = (Q-1)/2$ and $p \neq q$. Let N be a publicly known value.

Setup. Two random generators $g_p \leftarrow \mathbb{G}_p$ and $g_q \leftarrow \mathbb{G}_q$ are found, so that $g_p \neq 1$ and $g_q \neq 1$. $\mathbb{G}_p$ and $\mathbb{G}_q$ are subgroups of the order p and q in $\mathbb{Z}_P^*$ and $\mathbb{Z}_Q^*$ respectively. Next $b_0 \in \mathbb{Z}_N^*$ is computed using the Chinese Remainder Theorem so that $b_0 = g_p \mod P$ and $b_0 = g_q \mod Q$. The computed value b_0 is a generator element of $\mathbb{G}_{pq}$. Next an element α is uniformly chosen from $\mathbb{Z}_{pq}^*$ and the value b_1 is computed $b_1 = b_0^\alpha \mod N$. The generator algorithm Gen outputs the following values as the public key:

$$\mathsf{pk} = (b_0, b_1) \ .$$

Commitment. To commit to a value $m \in \mathbb{Z}_N$, the sender uniformly chooses $r \leftarrow \mathbb{Z}_{2^k N}$ and computes

$$\mathsf{Com}_{\mathsf{pk}}(m; r) = (b_0^m b_1^r \mod N, (m; r)) \ .$$

Opening. To open the commitment, the sender transmits m and r and the receiver can compute $b_0^m b_1^r \mod N$ to check whether the received commitment value is valid.

Theorem 5.3. *Let N be chosen uniformly from an RSA-modulus class that is (t, ε)-hard, then the Fujisaki-Okamoto commitment scheme is* (2^{-k})-hiding *and* $(\tau, \frac{\varepsilon}{8})$-binding, *where $\tau = t - \mathcal{O}(1)$.*

Proof. HIDING. First we show that the scheme is statistically hiding. For a moment assume that r is chosen from $\mathbb{Z}_{\varphi(N)}$, where $\varphi(\cdot)$ is the Eulerian function, then the scheme is perfectly hiding. The cardinality of the subgroup generated by b_0 divides the cardinality of $\mathbb{Z}_N^*$ and as $|\mathbb{Z}_N^*| = \varphi(N)$, then $|\langle b_0 \rangle|$ divides $\varphi(N)$. We know that the order of $\langle b_0 \rangle$ is pq, and that $\gcd(\alpha, pq) = 1$, because α is chosen from $\mathbb{Z}_{pq}^*$. Then $b_0^\alpha = b_1$ is a generator of $\mathbb{G}_p$ and $\mathbb{G}_q$, hence, the order of b_1 is equal

to the order of b_0, implying that $\langle b_0 \rangle = \langle b_1 \rangle$. This means that $|\langle b_0 \rangle|$ divides $\varphi(N)$ and, thus, if r is taken uniformly from $\mathbb{Z}_{\varphi(N)}$, then b_1^r has uniform distribution over $\langle b_0 \rangle$ and, thus, $b_0^m b_1^r$ is indistinguishable from a uniformly chosen element of $\langle b_0 \rangle$.

However, when we know $\varphi(N)$, then we can efficiently find the factorisation of N, hence, we have to take r from a set $\mathbb{Z}_{2^k N}$ that achieves a similar distribution as $\mathbb{Z}_{\varphi(N)}$. We know that $N = PQ = (2p+1)(2q+1) = 4pq + 2p + 2q + 2$ and that $\varphi(N) = \varphi(PQ) = (P-1)(Q-1) = 4pq$, so we can say that, although N and $\varphi(N)$ are fairly similar values, $\varphi(N)$ does not divide $2^k N$ and, thus, the scheme can achieve at most statistical hiding. It suffices to show that the statistical difference between $\mathbb{Z}_{\varphi(N)}$ and the distribution created by taking elements from $\mathbb{Z}_{2^k N}$ modulo $\varphi(N)$ is negligible. We use the bound from Lemma 2.1 so that $T = 2^k N$ and $u = \varphi(N)$ and we see that the statistical difference is

$$\varepsilon \leq \frac{\varphi(N)}{2^k N} = \frac{4pq}{2^k(4pq + 2p + 2q + 2)} \leq \frac{1}{2^k} \ .$$

This value, however, is negligible if k is sufficiently large and it is almost impossible to tell the difference between the two distributions, hence, if we substitute $\mathbb{Z}_{\varphi N}$ with $\mathbb{Z}_{2^k N}$, then the initial zero advantage of the adversary increases by at most 2^{-k}.

Binding. To prove that the scheme is computationally binding we show how to convert a τ-time adversary A against the binding property to an adversary B that can factorise N. Assume that A achieves advantage

$$\mathrm{Adv}^{\mathsf{bind}}(A) = \Pr\left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}, (c, d_0, d_1) \leftarrow A(\mathsf{pk}) : \\ \perp \neq \mathsf{Open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{Open}_{\mathsf{pk}}(c, d_1) \neq \perp \end{array}\right] \leq \varepsilon \ .$$

Our task is now to find an adversary B that uses A to find the the factorisation of N. If A outputs a valid double opening then $\mathsf{Com}_{\mathsf{pk}}(m_0; r_0) = \mathsf{Com}_{\mathsf{pk}}(m_1; r_1)$ then B computes $m = m_0 - m_1$ and $r = r_0 - r_1$ and has, thus, found such values that

$$b_0^m b_1^r \equiv 1 \mod N \ . \tag{6}$$

The adversary B can use these values to find a multiple of $\varphi(N)$ and thus also find the factorisation of N. From congruence (6), we get

$$\begin{aligned} b_0^m b_0^{\alpha r} &\equiv b_0^{\varphi(N)} \mod N \\ m + \alpha r &\equiv 0 \mod \varphi(N) \\ m + \alpha r &= t\varphi(N) \ . \end{aligned}$$

As we know the values of m, r and α, we can compute the value of $t\varphi(N)$. From Lemma 2.3 we see that it is possible to find the factorisation of N given a multiple of $\varphi(N)$. Now, consider an adversary B that acts as a challenger in the binding game and uses the adversary A for finding a double opening

$B(N)$

$b_0 \leftarrow \mathbb{Z}_N$ \\ to get b_0 from $\mathbb{Z}_{pq}^*$

$\alpha \leftarrow \mathbb{Z}_{2^k N}$ \\ to get α from $\mathbb{Z}_{pq}^*$

$b_1 \leftarrow b_0^{\alpha}$

$(c, (m_0; r_0), (m_1; r_1)) \leftarrow A(b_0, b_1)$

$m \leftarrow m_0 - m_1$

$r \leftarrow r_0 - r_1$

$x \leftarrow \mathbb{Z}_N$

Return $\gcd(x, N)$, if $\gcd(x, N) \neq 1$

if $x^a = \pm 1$, then halt

else Return $\gcd(x^a + 1, N)$

The probability that the order of b_0 chosen by B is pq is non-negligible, because

$$\frac{\varphi(pq)}{|\mathbb{Z}_N|} = \frac{(p-1)(q-1)}{(2p+1)(2q+1)} \approx \frac{1}{4} \ .$$

The algorithm succeeds in factorising N with a probability not less than $\frac{1}{8}$, provided that A outputs a valid double opening, as there is a chance of not less than $\frac{1}{2}$ that he is able to find $x^a \neq \pm 1$. Hence, the advantage of the adversary B is approximately $\frac{1}{8}\,\mathrm{Adv}(A)$. It should be noted that the adversary can repeat the search for a suitable b_0 and x, until he is able to factorise N and, thus, raise the probability of success arbitrarily close to $\mathrm{Adv}(A)$. To conclude the proof, observe that the working time of B is $t = \tau + \mathcal{O}(1)$. □

5.3 Trapdoor Discrete Logarithm

We show how to find the trapdoors in the Pedersen and Fujisaki-Okamoto commitments that are both perfectly equivocable trapdoor commitment schemes. The trapdoor is made up of all the values necessary for altering the decommitment as the trapdoor.

Theorem 5.4. *The Pedersen commitment scheme is equivocable.*

Construction. First we run Gen^* that outputs the public key $\mathsf{pk} = (g, y)$ and the secret key $\mathsf{sk} = x$, $x \in \mathbb{Z}_q$ so that $y = g^x \mod q$. The equivocability $\mathsf{Equiv}_{\mathsf{sk}}$ function creates the double opening to a given commitment $c = g^{m_0} y^{r_0}$. On input c and a message $m_1 \in \mathbb{Z}_q$ so that $m_1 \neq m_0$, $\mathsf{Equiv}_{\mathsf{sk}}$ computes the new randomness

$$r_1 = r_0 + (m_0 - m_1)x^{-1} \mod q.$$

and outputs the new decommitment value $d_1 = (m_1; r_1)$.

It remains to show that d_1 is a valid decommitment value for the commitment c. It suffices to show that the commitments of both message-randomness pairs are equal. We know that in the Pedersen commitment scheme $\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) = (g^{m_1} y^{r_1}, (m_1; r_1))$, now consider the following equation

$$\begin{aligned} g^{m_1} y^{r_1} &= g^{m_1} y^{r_0 + (m_0 - m_1)x^{-1}} &&= g^{m_1} y^{r_0} y^{(m_0 - m_1)x^{-1}} &&= g^{m_1} y^{r_0} (g^x)^{(m_0 - m_1)x^{-1}} \\ &= g^{m_1} y^{r_0} g^{x(m_0 - m_1)x^{-1}} &&= g^{m_1} y^{r_0} g^{m_0 - m_1} &&= g^{m_1 + m_0 - m_1} y^{r_0} \\ &= g^{m_0} y^{r_0}. \end{aligned}$$

It is straightforward to see that this is equal to the commitment c. In addition, we can see that r_1 has uniform distribution over $\mathbb{Z}_q$, because r_0 is chosen uniformly from that group. This means that the false decommitment has the same distribution that a real decommitment would have.

Theorem 5.5. *The Fujisaki-Okamoto commitment scheme is equivocable.*

Construction. First we run Gen^* that outputs the public key $\mathsf{pk} = (b_0, b_1)$ and the secret key $\mathsf{sk} = \alpha$, $\alpha \in \mathbb{Z}^*_{pq}$ so that $b_1 = b_0^{\alpha} \mod N$. The equivocability $\mathsf{Equiv}_{\mathsf{sk}}$ function creates the double opening to a given commitment $c = b_0^{m_0} b_1^{r_0} \mod N$. On input c and a new message $m_1 \in \mathbb{Z}_n$ so that $m_1 \neq m_0$, $\mathsf{Equiv}_{\mathsf{sk}}$ computes the new randomness

$$r_1 = r_0 + (m_0 - m_1)x^{-1} \mod N.$$

and outputs the new decommitment value $d_1 = (m_1; r_1)$.

It remains to show that d_1 is a valid decommitment value for the commitment c. It suffices to show that the commitments of both message-randomness pairs are equal. We know that in the Fujisaki-Okamoto commitment scheme $\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) = (b_0^{m_1} b_1^{r_1}, (m_1; r_1))$, now consider the following equation

$$\begin{aligned} b_0^{m_1} b_1^{r_1} &= b_0^{m_1} b_1^{r_0 + (m_0 - m_1)x^{-1}} &&= b_0^{m_1} b_1^{r_0} b_1^{(m_0 - m_1)x^{-1}} &&= b_0^{m_1} b_1^{r_0} (b_0^x)^{(m_0 - m_1)x^{-1}} \\ &= b_0^{m_1} b_1^{r_0} b_0^{x(m_0 - m_1)x^{-1}} &&= b_0^{m_1} b_1^{r_0} b_0^{m_0 - m_1} &&= b_0^{m_1 + m_0 - m_1} b_1^{r_0} \\ &= b_0^{m_0} b_1^{r_0} \ . \end{aligned}$$

It is straightforward to see that this is equal to the commitment c. In addition, we can see that r_1 has uniform distribution over $\mathbb{Z}_N$, because r_0 is chosen uniformly from that group. This means that the false decommitment has the same distribution that a real decommitment would have.

6 Commitments and Non-Malleability

Usually, when talking about different protocols, we think about two parties exchanging information. But there is always a chance that a malicious party might listen to or even interfere with the communication—this is known as the man-in-the-middle attack. The most classical example of this attack concerns ballot boxes. When the votes have been cast and the malicious party gains access to the ballot box, it is very easy for him to include his votes in the box, thus altering the results to his advantage. Note that this does not require the adversary to break the hiding or binding property, just adding a related message is sufficient. This kind of attack is a threat to commitment schemes as well—malleability allows an adversary *Ed* to alter a commitment received from *Alice*, in a meaningful way so that the receiver *Bob* cannot make sure whether the commitment is original or it has been tampered with (Figure 7).

$$Alice \xrightarrow{x} Ed \xrightarrow{x+y} Bob$$

Figure 7: Man-in-the-middle attack

Non-malleability is a property that prevents an adversary from making meaningful changes to the messages being passed from one party to the other. Non-malleability with respect to commitment denies the adversary the possibility to create a new commitment from an existing one, whereas non-malleability with respect to opening allows the adversary to make a commitment but not open it. The difference between non-malleability with respect to commitment and with respect to opening was first defined in the article [FF00]. We give both of the definitions here. Although non-malleabiliy w.r.t. commitment is a stronger notion, non-malleability w.r.t. opening has often been considered enough for all practical applications [FF00]. In the following, when we talk about non-malleability, we mean non-malleability w.r.t. commitment, if not specified otherwise.

We give the descriptions of the two non-malleability games in figures. Non-malleability w.r.t. opening can be seen in Fig. 8 and non-malleability w.r.t. commitment is given in Fig. 9. In both of the games, the adversary has to decide which message the commitment was made for. The two games begin similarly—first the key generation algorithm Gen is run to produce the public key pk and the A_1 part of the adversary outputs two messages m_0, m_1 and an internal state σ_1. Next the challenger uniformly chooses a bit s and creates a commitment-decommitment pair for message m_s. The commitment value c from this pair is given along with σ_1 to A_2 that outputs a tuple of commitments $(\hat{c}_1, \ldots, \hat{c}_n)$ and σ_2. At this point the two non-malleability games go their separate ways.

$\mathcal{G}^A_{\mathsf{nm-open}}$

$\mathsf{pk} \leftarrow \mathsf{Gen}$

$(m_0, m_1, \sigma_1) \leftarrow A_1(\mathsf{pk})$

$s \leftarrow \{0,1\}$

$(c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s)$

$(\hat{c}_1, \ldots, \hat{c}_n, \sigma_2) \leftarrow A_2(c, \sigma_1)$

$(\hat{d}_1, \ldots, \hat{d}_n) \leftarrow A_3(d, \sigma_2)$

$y_i \leftarrow \mathsf{Open}_{\mathsf{pk}}(\hat{c}_i, \hat{d}_i), i = (1, \ldots, n)$

halt if $c \in (\hat{c}_1, \ldots, \hat{c}_n) \vee \perp \in (y_1, \ldots, y_n)$

if $A_4(m_1, y_1, \ldots, y_n, \sigma_2) = s$, return 1

else return 0

Figure 8: The game for non-malleability w.r.t. opening

Non-malleability w.r.t. opening means that an adversary, given a commitment is not able to create a correct related commitment that he is able to open himself. In this case the tuple created by A_2 is given along with the decommitment value d and the state σ_2 to A_3 that outputs a tuple of decommitment values $(\hat{d}_1, \ldots, \hat{d}_n)$. The commitments $(\hat{c}_1, \ldots, \hat{c}_n)$ are opened using the decommitments, and a tuple $(y_1, \ldots, y_n)$ is received. If the original commitment c is in the tuple $(\hat{c}_1, \ldots, \hat{c}_n)$ or any of the opened commitments opened to $\perp$, the game is halted, otherwise, the game outputs the decision made by A_4 that is given the message m_1, the tuple $(y_1, \ldots, y_n)$ and the internal value σ_2 as input.

Non-malleability with respect to commitment means that given a commitment, an adversary is not able to create a correct related commitment that can be opened at all. In this case the tuple created by A_2 is given to the extraction oracle Extr that opens them and receives a tuple $(y_1, \ldots, y_n)$. If the original commitment c is in the tuple $(\hat{c}_1, \ldots, \hat{c}_n)$, the game is halted, otherwise, the game outputs the decision made by A_4 that is given the message m_1, the tuple $(y_1, \ldots, y_n)$ and the internal value σ_2 as input.

Definition 6.1. *A commitment scheme is* (t, ε)-non-malleable with respect to decommitment *if any* t*-time adversary* $A = (A_1, A_2, A_3, A_4)$ *playing the game* $\mathcal{G}_{\mathsf{nm-open}}$ *achieves advantage*

$$\mathrm{Adv}^{\mathsf{nm}}_{\mathcal{C}om}(A) = \left|2 \cdot \Pr\left[\mathcal{G}^A = s\right] - 1\right| \leq \varepsilon \ .$$

The following definition [BS99, FF00] is sensible only, if the commitment

$\mathcal{G}^A_{\mathsf{nm-com}}$

$\mathsf{pk} \leftarrow \mathsf{Gen}$
$(m_0, m_1, \sigma_1) \leftarrow A_1(\mathsf{pk})$
$s \leftarrow \{0, 1\}$
$(c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s)$
$(\hat{c}_1, \ldots, \hat{c}_n, \sigma_2) \leftarrow A_2(c, \sigma_1)$
$(y_1, \ldots, y_n) \leftarrow \mathsf{Extr}_{\mathsf{sk}}(\hat{c}_1, \ldots, \hat{c}_n)$
halt if $c \in (\hat{c}_1, \ldots, \hat{c}_n)$
if $A_4(m_1, y_1, \ldots, y_n, \sigma_2) = s$, return 1
else return 0

Figure 9: The game for non-malleability w.r.t. commitment

scheme is either statistically binding or extractable.

Definition 6.2. *A commitment scheme is* (t, ε)-non-malleable with respect to commitment *if any* t*-time adversary* $A = (A_1, A_2, A_4)$ *playing the game* $\mathcal{G}_{\mathsf{nm-com}}$ *achieves advantage*

$$\mathrm{Adv}^{\mathsf{nm}}_{\mathcal{C}om}(A) = \left|2 \cdot \Pr\left[\mathcal{G}^A = s\right] - 1\right| \leq \varepsilon \ ,$$

where Extr *is a computable function such that* $\mathsf{Extr}_{\mathsf{pk}}(c) = x$ *if* $(c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(x)$.

If the commitment scheme is extractable, we can use the secret key and the oracle $\mathsf{Extr}_{\mathsf{sk}}(\cdot)$ instead of $\mathsf{Extr}_{\mathsf{pk}}(\cdot)$. It is interesting to note that a commitment that is non-malleable with respect to commitment is also non-malleable with respect to opening. When the adversary in the game of non-malleability with respect to commitment has given the commitments to the challenger, he is no longer able to attack in any way, even if he gets to know a backdoor or gets infinite computing power. However, in the case of non-malleability with respect to opening, the adversary can influence the input of A_4 after it has given the commitments to the challenger. But, as mentioned before, non-malleability w.r.t. opening is usually enough in practical applications.

Next, we will show that non-malleability implies hiding and binding. It suffices to show that this is true for the non-malleability property with respect to opening. Non-malleability with respect to commitment implies non-malleability with respect to opening, so it also implies hiding and binding because implication is transitive.

Theorem 6.1. *A commitment scheme that is* (t, ε)*-non-malleable with respect to opening is also* (τ, ε)*-hiding, where* $\tau = t - \mathcal{O}(1)$.

Proof. We use proof by contradiction to show that non-malleability w.r.t. opening implies hiding. For the sake of contradiction, assume that the τ-time adversary $B = (B_1, B_2)$ playing the hiding game, achieves advantage

$$\mathrm{Adv}^{\mathsf{hid}}_{\mathsf{Com}}(B) = \left| 2 \cdot \Pr \left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}, s \leftarrow \{0,1\}, (m_0, m_1, \sigma) \leftarrow B_1(\mathsf{pk}), \\ (c, d) \leftarrow \mathsf{Com}_{\mathsf{pk}}(m_s) : B_2(\sigma, c) = s \end{array} \right] - 1 \right| > \varepsilon \ .$$

Then the adversary $A = (A_1, A_2, A_3, A_4)$ can use the adversary B to win the non-malleability game depicted in Fig. 6.1. At the beginning of the game, A_1 uses B_1 to output (m_0, m_1, σ). The commitment c of one of these messages and the inner state σ are given to A_2 that passes them on to B_2. Similarly to the hiding game, B_2 now outputs a guess s and succeeds with probability greater than ε. This guess is put into the state σ_2. Everything works as before, except when A_4 receives σ_2 as part of its input, it no longer needs to output a guess itself, but can simply use the guess from B_2. So the adversary A also succeeds with probability greater than ε, but this contradicts the assumption that the scheme is non-malleable with respect to opening. □

Theorem 6.2. *A commitment scheme that is (t, ε)-non-malleable with respect to opening is also (τ, ε)-binding, where $\tau = t - \mathcal{O}(1)$.*

Proof. We use proof by contradiction to show that non-malleability w.r.t. opening implies binding. For the sake of contradiction, assume that the τ-time adversary B playing the binding game, achieves advantage

$$\mathrm{Adv}^{\mathsf{bind}}_{\mathsf{Com}}(B) = \Pr \left[\begin{array}{l} \mathsf{pk} \leftarrow \mathsf{Gen}, (\hat{c}, \hat{d}_0, \hat{d}_1) \leftarrow B(\mathsf{pk}) : \\ \bot \neq \mathsf{Open}_{\mathsf{pk}}(\hat{c}, \hat{d}_0) \neq \mathsf{Open}_{\mathsf{pk}}(\hat{c}, \hat{d}_1) \neq \bot \end{array} \right] > \varepsilon \ .$$

Then the adversary $A = (A_1, A_2, A_3, A_4)$ can use the adversary B to win the non-malleability game from Fig. 6.1. The A_1 part of the adversary outputs two messages m_0, m_1 and a state σ_1. Next, A_2 can use B to create a commitment $\hat{c}$ and find a double decommitment $(\hat{d}_0, \hat{d}_1)$ that opens $\hat{c}$ to either y_0 or y_1. The new commitment $\hat{c}$ and the decommitment pair $(\hat{d}_0, \hat{d}_1)$ are enclosed in σ_2. We let A_2 output $\hat{c}, \sigma_2$ when it receives c and σ_1. We know that by definition A_3 has access to the original decommitment value d, but the problem is that it cannot pass on any implicit information to A_4 about d or the message that was committed to. With the help of the adversary B we now have a situation, where A_3 knows which of the two messages the commitment c belongs to, and it is also capable of forwarding the information about this one bit to A_4. To do this, A_3 simply chooses the corresponding decommitment from the pair $(\hat{d}_0, \hat{d}_1)$—it chooses $\hat{d}_0$ when the commitment opens to m_0, and $\hat{d}_1$ otherwise.

It is quite straightforward to see that when the function Open is run on the commitment from A_2 and decommitment from A_3, the result y is either y_0 or y_1. This means that y represents the index of the message that the original commitment c was made to. Now, it is simple for A_4 to output the correct answer. The adversary A succeeds with the same probability as B. This probability, however, is larger than ε and this contradicts the assumption that the scheme is non-malleable with respect to opening. □

6.1 Homomorphic Commitment Schemes

Not all schemes are non-malleable and, to illustrate this, we discuss homomorphic commitments. Schemes with this property are malleable, i.e., they allow transformations on the commitment to produce meaningful changes in the message being committed to. Even though this leaves the commitment vulnerable to man-in-the-middle attacks, homomorphism is not a bad property and can be used in different operations connected with commitment schemes.

In the following definition, $\circ$ denotes any efficiently computable binary operation on commitment-decommitment pairs. In addition, we require that the value $\mathsf{Com}_{\mathsf{pk}}(m_1+m_2)$ be made from the values c_1 and c_2, without having seen the corresponding decommitment values d_1 and d_2, because in practice the decommitment values are initially not revealed.

Definition 6.3. *A commitment scheme is homomorphic, if*

$$\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) \circ \mathsf{Com}_{\mathsf{pk}}(m_2; r_2) = \mathsf{Com}_{\mathsf{pk}}(m_1 + m_2; r_1 + r_2)$$

and the distributions of $\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) \circ \mathsf{Com}_{\mathsf{pk}}(m_2; r_2)$ *and* $\mathsf{Com}_{\mathsf{pk}}(m_1+m_2; r_1+r_2)$ *coincide even if one of the initial commitments is fixed.*

We show that the three commitment schemes—Pedersen, Fujisaki-Okamoto and ElGamal schemes—based on the discrete logarithm problem are all homomorphic. For the Pedersen and Fujisaki-Okamoto schemes we define the operand $\circ$ for the commitment-decommitment pairs in the following way

$$(c_1, d_1) \circ (c_2, d_2) = (c_1 \cdot c_2, d_1 + d_2) \ .$$

Theorem 6.3. *The Pedersen commitment scheme is homomorphic.*

Proof. In the case of the Pedersen commitment scheme we know that $\mathsf{Com}_{\mathsf{pk}}(m; r) = (g^m y^r, (m; r))$. To show that it is homomorphic we multiply two commitments and add the corresponding decommitments to each other. Consider the equation

$$\begin{aligned} \mathsf{Com}_{\mathsf{pk}}(m_1; r_1) \circ \mathsf{Com}_{\mathsf{pk}}(m_2; r_2) &= (g^{m_1} y^{r_1} g^{m_2} y^{r_2}, (m_1 + m_2; r_1 + r_2)) \\ &= (g^{m_1+m_2} y^{r_1+r_2}, (m_1 + m_2; r_1 + r_2)) \\ &= \mathsf{Com}_{\mathsf{pk}}(m_1 + m_2; r_1 + r_2) \ . \end{aligned}$$

It is clear that this is the commitment to the message $m_1 + m_2$. □

Theorem 6.4. *The Fujisaki-Okamoto commitment scheme is homomorphic.*

Proof. We know that in the Fujisaki-Okamoto commitment scheme $\mathsf{Com}_{\mathsf{pk}}(m; r) = (b_0^m b_1^r, (m; r))$. Similarly to the last proof, we multiply two commitments and add the corresponding decommitments to each other. Consider the following equation

$$\begin{aligned}\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) \circ \mathsf{Com}_{\mathsf{pk}}(m_2; r_2) &= (b_0^{m_1} b_1^{r_1} b_0^{m_2} b_1^{r_2}, (m_1 + m_2; r_1 + r_2)) \\ &= (b_0^{m_1+m_2} b_1^{r_1+r_2}, (m_1 + m_2; r_1 + r_2)) \\ &= \mathsf{Com}_{\mathsf{pk}}(m_1 + m_2; r_1 + r_2) \ .\end{aligned}$$

It is clear that this is the commitment to the message $m_1 + m_2$. □

We define the operand $\circ$ differently for the ElGamal scheme, because the commitment consists of a pair of values. Let us assume that we have two commitment-decommitment pairs $((c_{11}, c_{12}), d_1)$ and $((c_{21}, c_{22}), d_2)$. Then

$$((c_{11}, c_{12}), d_1) \circ ((c_{21}, c_{22}), d_2) = ((c_{11} \cdot c_{21}, c_{12} \cdot c_{22}), d_1 + d_2) \ .$$

Theorem 6.5. *The ElGamal commitment scheme is homomorphic.*

Proof. We take commitment-decommitment pairs for the messages m_1 and m_2: $\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) = ((g^{r_1}, g^{m_1} g^{ar_1}), (m_1; r_1))$, $\mathsf{Com}_{\mathsf{pk}}(m_2; r_2) = ((g^{r_2}, g^{m_2} g^{ar_2}), (m_2; r_2))$ and do the following computations

$$\begin{aligned}\mathsf{Com}_{\mathsf{pk}}(m_1; r_1) \circ \mathsf{Com}_{\mathsf{pk}}(m_2; r_2) &= ((g^{r_1} g^{r_2}, g^{m_1} g^{ar_1} g^{m_2} g^{ar_2}), (m_1 + m_2; r_1 + r_2)) \\ &= ((g^{r_1+r_2}, g^{m_1+m_2} g^{a(r_1+r_2)}), (m_1 + m_2; r_1 + r_2)) \\ &= \mathsf{Com}_{\mathsf{pk}}(m_1 + m_2; r_1 + r_2) \ .\end{aligned}$$

This is the commitment of the message $m_1 + m_2$. □

Homomorphic commitment can be used to perform different operations with commitments. An intuitive example is a protocol with $n > 2$ parties and one trusted party, where all parties send commitments $c_1, \ldots, c_n$ to messages $m_1, \ldots, m_n$ to the trusted party that uses the binary operation denoted by $\circ$ to compute the commitment c of the sum m of the messages. The trusted party can publish c and, if the parties wish to know the sum, they send the decommitments $d_1, \ldots, d_n$ to the trusted party that computes the common decommitment value d. When the trusted party publishes this value, all of the parties will know the sum of the messages, but they will not get any idea about the individual messages of other parties. This is the underlying idea of many e-voting systems.

6.2 IND-CCA2 Security and Non-Malleability

In this subsection, we show that IND-CCA2 security (defined in Section 4.1) implies non-malleability. As discussed before, the commitment scheme needs to be extractable to achieve duality with encryption schemes. In the following theorems, let $\mathcal{E}nc = (\mathsf{Gen}_{\mathcal{E}nc}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme and $\mathcal{C}om = (\mathsf{Gen}_{\mathcal{C}om}, \mathsf{Gen}^*_{\mathcal{C}om}, \mathsf{Com}, \mathsf{Open}, \mathsf{Extr})$ be a commitment scheme.

Theorem 6.6. *Let $\mathcal{E}nc$ and $\mathcal{C}om$ be in canonical correspondence. Then (t, ε)-IND-CCA2 security implies (t, ε)-non-malleability with respect to commitment.*

Proof. Since $\mathcal{E}nc$ and $\mathcal{C}om$ are in canonical correspondence, we can unify the extraction and decryption oracles as $\mathsf{Extr}_{\mathsf{sk}}(\cdot)$. We use proof by contradiction to show that the constructed commitment scheme $\mathcal{C}om$ is non-malleable. For the sake of contradiction, we assume that, the encryption scheme is IND-CCA2 secure but the commitment scheme is not non-malleable. Let $A = (A_1, A_2, A_4)$ be a corresponding adversary that plays the game $\mathcal{G}^A_{\mathsf{nm-com}}$ from Fig. 9 and achieves advantage

$$\mathrm{Adv}^{\mathsf{nm}}_{\mathcal{C}om}(A) = \left|2 \cdot \Pr\left[\mathcal{G}^A = s\right] - 1\right| > \varepsilon \ .$$

Now we show that the scheme cannot be IND-CCA2 secure. We use the adversary A to construct another adversary $B = (B_1, B_2)$, where B_1 is the same as A_1 receiving pk and outputting (m_0, m_1, σ_1) and B_2 is constructed by uniting parts A_2 and A_4 of the adversary A. In the IND-CCA2 game B_2 gets the encryption e and additional information from B_1 in the form of σ_1. This state contains, without loss of generality, the messages m_0 and m_1. By definition, B_2 can use the decryption oracle and ask it to decrypt any encryption he wants except the one he was given, and then has to output its guess about which of the messages was encrypted. We can use parts of the adversary A to execute the described actions. The input for B_2 is given to A_2 that outputs the tuple of commitments $(\hat{c}_1, \ldots, \hat{c}_n)$ that will be given as a query to the extraction oracle $\mathsf{Extr}_{\mathsf{sk}}(\cdot)$, and an internal state σ_2. As mentioned before, the extraction and decryption oracles coincide because of the correspondence between $\mathcal{E}nc$ and $\mathcal{C}om$ and, thus, B_2 can submit the query to $\mathsf{Extr}_{\mathsf{sk}}(\cdot)$ in the IND-CCA2 game. The oracle outputs the tuple of results $(y_1, \ldots, y_n)$ that are given as input to A_4. Now, the output s' of A_4 is also given as the output of B_2. The algorithm for B_2 can compactly be written as

$B_2(e, \sigma_1)$
$\left[\begin{array}{l} (\hat{c}_1, \ldots, \hat{c}_n, \sigma_2) \leftarrow A_2(e, \sigma_1) \\ (y_1, \ldots, y_n) \leftarrow \mathsf{Extr}_{\mathsf{sk}}(\hat{c}_1, \ldots, \hat{c}_n) \\ \mathsf{Return}\ A_4(m_1, y_1, \ldots, y_n, \sigma_2) \end{array}\right.$

The adversary B plays the IND-CCA2 game. Since B by construction behaves exactly like A, it achieves advantage

$$\mathrm{Adv}^{\mathsf{ind-cca2}}(B) = \left| 2 \cdot \Pr \left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}, s \leftarrow \{0,1\}, \\ (m_0, m_1, \sigma_1) \leftarrow B_1(\mathsf{pk}), \\ e \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_s) : B_2^{\mathsf{Extr}_{\mathsf{sk}}(\cdot)}(\sigma_1, e) = s \end{array} \right] - 1 \right| > \varepsilon \ .$$

But this contradicts the assumption that the encryption scheme is IND-CCA2 secure. Hence, the commitment scheme must be non-malleable. □

Unfortunately, non-malleability under chosen plaintext attack (NM-CPA) does not imply IND-CCA2 security. On the other hand, it has been proved that non-malleability under chosen ciphertext attack (NM-CCA2) implies IND-CCA2 security and *vice versa* [BS99, DDN91].

It is possible to transform any given equivocable commitment scheme into a commitment scheme that is non-malleable with respect to decommitment. This construction is given in the article [CIO98]. Moreover, in [Cre02] Crescenzo shows that it is possible to construct a commitment scheme that is non-malleable with respect to commitment from an extractable and equivocable commitment scheme.

6.3 An Example of Non-Malleable Commitment

We look at the Cramer-Shoup encryption scheme [CS98] and construct a non-malleable commitment scheme from it, using the canonical correspondence between encryptions and commitment. The Cramer-Shoup cryptosystem is based on the assumptions that the DDH problem is hard and there exists a collision resistant hash function. The constructed commitment scheme is provably non-malleable and it is quite straightforward to see that there exists a way to verify whether the commitment has been tampered with.

6.3.1 Encryption Scheme

Setup. Let $\mathbb{G}_q$ be a group of prime order q and let H be a hash function that hashes long strings to elements of $\mathbb{Z}_q$. Random elements $g_1, g_2 \leftarrow \mathbb{G}_q$ and $x_1, x_2, y_1, y_2, z_1, z_2 \leftarrow Z_q$ are chosen. Next, three group elements

$$c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^{z_1} g_2^{z_2}$$

are computed. The values (g_1, g_2, c, d, h) form a public key, the secret key is $(x_1, x_2, y_1, y_2, z_1, z_2)$.

Encryption. To encrypt a message $m \in \mathbb{G}_q$, the encryption algorithm chooses a random $r \leftarrow \mathbb{Z}_q$ and computes

$$u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = \mathsf{H}(u_1, u_2, e), v = c^r d^{r\alpha} \ .$$

The encryption is

$$\mathsf{Enc}(m; r) = (u_1, u_2, e, v) = (g_1^r, g_2^r, h^r m, c^r d^{r\alpha}) \ .$$

Decryption. Given an encryption (u_1, u_2, e, v), the decryption algorithm first computes $\alpha = \mathsf{H}(u_1, u_2, e)$ and tests if

$$u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = v \ .$$

If this condition does not hold, the algorithm rejects the encryption. This is to make sure that the encryption has not been tampered with. Otherwise it outputs

$$m = e/(u_1^{z_1} u_2^{z_2}) \ .$$

Correctness. Correctness means that, if both parties are honest, it is possible to correctly decrypt the encryption and get the right message. The verification step in the decryption phase is correct:

$$u_1^{x_1} u_2^{x_2} (u_1^{y_1} u_2^{y_2})^\alpha = g_1^{rx_1} g_2^{rx_2} (g_1^{ry_1} g_2^{ry_2})^\alpha = (g_1^{x_1} g_2^{x_2})^r (g_1^{y_1} g_2^{y_2})^{r\alpha} = c^r d^{r\alpha} = v \ .$$

The decryption itself is also correct:

$$e/(u_1^{z_1} u_2^{z_2}) = e/(g_1^{rz_1} g_2^{rz_2}) = e/(g_1^{z_1} g_2^{z_2})^r = (h^r m)/h^r = m \ .$$

Therefore, the test performed by the decryption algorithm passes and the message will be opened correctly.

6.3.2 Commitment Scheme

We derive a commitment scheme from the given encryption scheme by canonical correspondence. We need three functions Gen, Com and Open. The key generation function of the commitment scheme outputs the public key from the key pair $((g_1, g_2, c, d, h), (x_1, x_2, y_1, y_2, z_1, z_2))$ from the key generation function of the encryption scheme. Since we do not need the secret key at all, it is simpler to generate the public key in the following way, as all values g_1, g_2, c, d and h are elements with a uniform distribution over $\mathbb{G}_q$.

Setup. Let $\mathbb{G}_q$ be a group of prime order q and let H be a hash function that hashes long strings to elements of $\mathbb{Z}_q$. Elements g_1, g_2, c, d, y are uniformly chosen from $\mathbb{G}_q$. The values g_1, g_2, c, d, y are published

$$(g_1, g_2, c, d, y) \leftarrow \mathsf{Gen} \ .$$

The commitment function outputs the encryption of the message as the commitment part and the message-randomness pair as the decommitment part of the commitment-decommitment pair.

Commitment. To commit to a message $m \in \mathbb{G}_q$, the sender chooses a random $r \leftarrow \mathbb{Z}_q$ and computes

$$u_1 = g_1^r, u_2 = g_2^r, e = y^r m, \alpha = \mathsf{H}(u_1, u_2, e), v = c^r d^{r\alpha} \ .$$

The commitment is

$$\mathsf{Com}_{\mathsf{pk}}(m; r) = ((u_1, u_2, e, v), (m; r)) = ((g_1^r, g_2^r, y^r m, c^r d^{r\alpha}), (m; r)) \ .$$

Opening. The sender sends the message m and the random r to the receiver. The receiver first computes $\alpha = \mathsf{H}(u_1, u_2, e)$ and tests if

$$c^r d^{r\alpha} = v \ .$$

If this condition does not hold, the algorithm rejects the commitment. This is to make sure that the commitment has not been tampered with. Otherwise the receiver checks whether the commitment is valid by using the commitment algorithm to compute it.

6.4 Simulation-Sound Trapdoor Commitments

We also briefly discuss simulation-sound trapdoor schemes [GMY03] that can be used to construct various zero knowledge protocols. These schemes have the simulation-sound binding property or strong binding property. The equivocability oracle can be used, but not when playing the binding game. This means that the commitment made by the adversary is binding even though A can use the equivocability oracle $\mathsf{Equiv}(\cdot)$. The definition uses the alternative key generation function Gen^* that outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

Definition 6.4. *A commitment scheme is a* (t, ε)-simulation-sound trapdoor scheme *if any* t*-time adversary* A *achieves advantage*

$$\mathrm{Adv}(A) = \Pr\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}^*, (c, d_0, d_1) \leftarrow A^{\mathsf{Equiv}(\cdot)}(\mathsf{pk}) : \\ \perp \neq \mathsf{Open}_{\mathsf{pk}}(c, d_0) \neq \mathsf{Open}_{\mathsf{pk}}(c, d_1) \neq \perp\end{array}\right] \leq \varepsilon \ .$$

The distributions of the public keys output by Gen *and* Gen^* *coincide. The equivocability oracle* $\mathsf{Equiv}(\cdot)$ *consists of two parts* O_1 *and* O_2 *that work in the following way:*

- *when asked to* commit, $\mathsf{Equiv}(\cdot)$ *outputs a fake commitment* $(\hat{c}, \sigma) \leftarrow \mathsf{Com}^*_{\mathsf{sk}}$ *and stores the pair* $(\hat{c}, \sigma)$,
- *when asked to* decommit *and given a fake commitment* $\hat{c}$ *and a message* m, $\mathsf{Equiv}(\cdot)$ *finds the stored* σ *and outputs a decommitment value* $\hat{d} \leftarrow \mathsf{Equiv}_{\mathsf{sk}}(m, \hat{c}, \sigma)$.

The adversary can only once ask the oracle to create a decommitment value for the commitment c *being double opened.*

Note that the definition of the strong binding property is very similar to the binding property of commitment schemes with the addition of the Equiv oracle—it describes a nontrivial attack against binding with the possibility to use the Equiv oracle. We also note that the IND-CCA2 property is similarly dual with a nontrivial attack against the hiding property with the possibility to use the Extr oracle. We have already proved that the IND-CCA2 property implies non-malleability w.r.t. commitment and it is possible to show that the strong binding property implies non-malleability w.r.t. opening. Moreover, in the article [MY04] the authors show that the strong binding property is equivalent with non-malleability with respect to opening. Recall that the IND-CCA2 property is equivalent to the non-malleability w.r.t. commitment property that uses a CCA2 oracle. One cannot help noticing that non-malleability is closely related to security under CCA2 attacks, i.e., the adversary can ask an oracle to open commitments during the attack against hiding or binding.

7 Conclusion

This thesis gives a classification of commitment schemes based on their properties. So far, there has been no such systematisation available in widespread cryptographic literature which is unfortunate because commitment schemes are an integral part of cryptography. They can have different properties: hiding, binding, extractability, equivocability, homomorphism, non-malleability. We show how these relate to each other and to the properties of encryption schemes. To illustrate, we bring examples of different schemes that are based on different assumptions and have different sets of properties.

The idea of a commitment scheme is the following: a sender commits to a message and transfers the commitment to the receiver. When the sender is ready to open the commitment, he reveals the decommitment value to the receiver, who can now check whether the commitment was valid. These schemes have two basic properties: hiding and binding. The hiding property denies the receiver the possibility to distinguish between commitments to different messages, and the binding property denies the sender the possibility of changing the message in the commitment. We also give the descriptions and security proofs of two commitment schemes: the simplified Canetti-Fischlin scheme based on pseudorandom generators, and the Halevi-Micali scheme based on collision resistant hash functions. These assumptions—the existence of pseudorandom generators and collision resistant hash function families—are among the most basic in cryptography.

Some commitment schemes have special properties in addition to hiding and binding. One of these properties is extractability that allows a party to extract the message from the commitment, if it knows a certain secret value. We show that this property is enough for a canonical correspondence to exist between encryption and commitment schemes. We also prove that the IND-CPA security property is equivalent with the computational hiding property of commitment schemes. We illustrate the concept of canonical correspondence by ElGamal encryption and commitment and also give security proofs for the latter.

Equivocability is another special property of commitment schemes. This property means that there exists a trapdoor so that if the sender knows a certain secret value, he is able to create a fake commitment and open it to any message he chooses. We give the descriptions and security proofs of two perfectly equivocable commitment schemes: the Pedersen scheme and the Fujisaki-Okamoto scheme. We also give the construction of a trapdoor for both of these schemes.

The last special property discussed in this thesis is non-malleability. This property prevents a malicious party from making meaningful changes to the commitment that is being passed between honest parties. There are two forms of the non-malleability property: w.r.t. opening and w.r.t. commitment. Non-malleability w.r.t. opening means that given a commitment, the adversary is unable to create a

related commitment that he is able to open, whereas non-malleability w.r.t. commitment means that the adversary is not even able to create a related commitment. The latter property is stronger but the former is considered to be enough in practical applications. To show that not every scheme is non-malleable, we talk about homomorphic schemes, where it is fairly easy to create a commitment to a related message from a given commitment. We also show that IND-CCA2 security implies non-malleability w.r.t. commitment and we describe the Cramer-Shoup commitment scheme as an example of a non-malleable scheme.

This thesis is useful for both academic and scientific purposes, as it describes commitment schemes that are a very important building block of cryptographic protocols. It gives a concise overview of the topic and is therefore useful for both students and practicing cryptographers.

8 Kinnistusskeemide homoloogiline klassifikatsioon

Magistritöö (40 AP)

Liina Kamm

Sisukokkuvõte

Käesoleva magistritöö eesmärk on süstematiseerida kinnistusskeeme ja nende omadusi. Töö sisaldab omaduste definitsioone ja nende omavahelisi suhteid ning näidisskeeme ja nende turvatõestusi. Lisaks kirjeldatakse kinnistusskeemide ja nende omaduste vastavust krüpteerimisskeemidega.

Kinnistusskeemid on krüptograafia oluline osa, mille tööpõhimõte on järgmine: saatja kinnistab sõnumi ning edastab selle saajale. Kui saatja on valmis sõnumit avaldama, edastab ta avamisväärtuse ning saajal on võimalus kontrollida, kas algne kinnistus oli korrektne. Sellistel skeemidel on kaks põhiomadust: peitvus ja siduvus. Peitvus tagab saatja turvalisuse ega lase saajal aru saada, mis väärtus on kinnistuse sees. Siduvus tagab saaja turvalisuse ega lase saatjal muuta sõnumit kinnistuse sees. Omaduste illustreerimiseks kirjeldame lihtsustatud Canetti-Fischlini skeemi, mis põhineb pseudojuhuslikkuse generaatoril, ja Halevi-Micali skeemi, mis põhineb kollisioonikindlal räsifunktsioonil, ning tõestame nende turvalisuse.

Mõnedel kinnistusskeemidel on lisaks siduvusele ja peitvusele veel eriomadusi, millest üks lihtsamaid on eraldatavus. See omadus laseb igal osapoolel, kellel on ligipääs teatud salajasele väärtusele, eraldada sõnum kinnistusest. Me näitame, et selle omaduse olemasolu on piisav tingimus, et leiduks vastavus krüpteerimis- ja kinnistusskeemi vahel. Lisaks tõestame, et IND-CPA turvalisus on samaväärne kinnistusskeemide arvutusliku peitvuse omadusega. Näitena toome ära vastavuse ElGamali krüpteerimis- ja kinnistusskeemi vahel ning tõestame viimase turvalisuse.

Kui kinnistusskeemil leidub tagauks, mille teadmine annab saatjale võimaluse väljastada võltskinnistus ning avada see ükskõik milliseks sõnumiks, siis kutsutakse skeemi mitmetähenduslikuks. Me kirjeldame kaht sellise omadusega kinnistusskeemi: Pederseni ja Fujisaki-Okamoto' skeemi. Esitame ka turvatõestused ning toome ära tagaukse konstrueerimiseks vajaliku arvutuskäigu.

Viimane eriomadus, mida me käsitleme, on mittedeformeeritavus, mis ei lase ründajal teha ausate poolte vahel edastatavale kinnistusele sisukaid muudatusi. Mittedeformeeritavust vaadeldakse avamise ja kinnistamise suhtes. Kui skeem on mittedeformeeritav avamise suhtes, siis ei ole vastane võimeline looma ausa kinnistusega seotud kinnistust nii, et ta seda hiljem avada suudaks. Sama omadus kinnistuse suhtes tähendab, et vastane ei ole suuteline seotud kinnistust üldse looma. Viimane omadus on ilmselgelt tugevam, aga nõrgemat omadust peetakse praktilistes

rakendustes piisavaks. Näitamaks, et iga skeem ei ole mittedeformeeritav, räägime homomorfsetest skeemidest, mille puhul on suhteliselt lihtne luua etteantud kinnistuse põhjal uus sisukas kinnistus. Lisaks tõestame, et IND-CCA2 turvalisusest järeldub mittedeformeeritavus kinnistuse suhtes, ning kirjeldame Cramer-Shoupi kinnistusskeemi, mis on mittedeformeeritav.

Käesolevale magistritööle sarnast ülevaadet kinnistusskeemidest ja nende omadustest hetkel teadaolevalt krüptograafia-alases kirjanduses ei leidu. Töö väärtuseks on tema sobivus nii akadeemilisteks kui teaduslikeks eesmärkideks, kuna seda on võimalik kasutada nii õppematerjali kui teatmikuna.

Autor soovib avaldada tänu oma juhendajatele, kelle järjekindlus ja entusiasm olid väärtuslikuks panuseks käesoleva magistritöö valmimisel.

References

[BBS86] Lenore Blum, Manuel Blum, and Mike Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[Blu81] Manuel Blum. Coin Flipping by Telephone. In *Advances in Cryptology: A Report on CRYPTO '81*, pages 11–15, 1981.

[Bon98] Dan Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science*, volume 1423, pages 48–63, 1998.

[BS99] Mihir Bellare and Amit Sahai. Non-Malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 519–536. Springer, 1999.

[BSS99] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic Curves in Cryptography*. Number 265 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1999.

[CF01] Ran Canetti and Marc Fischlin. Universally Composable Commitments. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 19–40. Springer, 2001.

[CIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-Interactive and Non-Malleable Commitment. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 141–150, New York, NY, USA, 1998. ACM Press.

[Cre02] Giovanni Di Crescenzo. Equivocable and Extractable Commitment Schemes. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2002.

[CS98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. In

Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, pages 13–25, 1998.

[CW77] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions (Extended Abstract). In *Conference Record of the Ninth Annual ACM Symposium on Theory of Computing, 2-4 May 1977, Boulder, Colorado, USA*, pages 106–112. ACM, 1977.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-Malleable Cryptography (Extended Abstract). In *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing, 6-8 May 1991, New Orleans, Louisiana, USA*, pages 542–552. ACM, 1991.

[DG03] Ivan Damgård and Jens Groth. Non-Interactive and Reusable Non-Malleable Commitment Schemes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 426–437, New York, NY, USA, 2003. ACM Press.

[DH76] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, number 2(6), pages 644–654, 1976.

[EJ02] Patrik Ekdahl and Thomas Johansson. A New Version of the Stream Cipher SNOW. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2002.

[FF00] Marc Fischlin and Roger Fischlin. Efficient Non-Malleable Commitment Schemes. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 413–431, London, UK, 2000. Springer-Verlag.

[FO97] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference*, pages 16–30, 1997.

[Gam84] Taher El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *CRYPTO*, pages 10–18, 1984.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions (Extended Abstract). In *25th Annual Symposium on Foundations of Computer Science, 24-26 October 1984, Singer Island, Florida, USA*, pages 464–479. IEEE, 1984.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, 5-7 May 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[GMY03] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening Zero-Knowledge Protocols Using Signatures. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2003.

[Gol04] Oded Goldreich. *Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

[HM96] Shai Halevi and Silvio Micali. Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, pages 201–215, 1996.

[JN03] Antoine Joux and Kim Nguyen. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups. *J. Cryptology*, 16(4):239–247, 2003.

[LAN05] Sven Laur, N. Asokan, and Kaisa Nyberg. Efficient Mutual Data Authentication Using Manually Authenticated Strings. Cryptology ePrint Archive, Report 2005/424, 2005.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography, Fifth Edition*. CRC Press, 2001.

[MY04] Philip D. MacKenzie and Ke Yang. On Simulation-Sound Trapdoor Commitments. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 382–400. Springer, 2004.

[Nao89] Moni Naor. Bit Commitment Using Pseudo-Randomness. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 128–136. Springer, 1989.

[Nat01] National Institute of Standards and Technology (NIST). FIPS-197: The Advanced Encryption Standard. NIST webpage http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001.

[Ped91] Torben Pryds Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference*, pages 129–140, 1991.

[RS91] Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer, 1991.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[RSA78] Ronald Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM 21,2*, pages 120–126, 1978.

[SCP00] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and Sufficient Assumptions for Non-iterative Zero-Knowledge Proofs of Knowledge for All NP Relations. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *Automata, Languages and Programming, 27th International Colloquium, ICALP 2000, Geneva, Switzerland, July 9-15, 2000, Proceedings*, volume 1853 of *Lecture Notes in Computer Science*, pages 451–462. Springer, 2000.

[Sha71] Daniel Shanks. Class Number, a Theory of Factorization, and Genera. In *Proceedings of Symposia in Pure Mathematics*, volume 20, pages 415–440, 1971.

[Sti06] Douglas R. Stinson. *Cryptography - Theory and Practice, Third Edition.* Chapman & Hall/CRC, 2006.