Homomorphic Cryptosystems and their Applications

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
dem Fachbereich Mathematik
der Universität Dortmund vorgelegt

von

Dörte K. Rappe

2. August 2004

Erster Gutachter: Prof. E. Becker
Zweiter Gutachter: Prof. J. Patarin

to my parents

Acknowledgement

The work I present in this thesis would not have been possible without the support of several people whom I would like to thank here.

First of all I would like to thank my supervisor Prof. Eberhard Becker for supporting me all the time by his advise and by giving me the freedom to explore several opportunities.
I am grateful to my thesis committee and to Prof. Jacques Patarin for being my external referee, for inviting me to continue my research at the University of Versailles and for offering me funding.
I owe thanks to my graduate school "Mathematische und ingenieurwissenschaftliche Methoden für sichere Datenübertragung und Informationsvermittlung" and the Deutsche Forschungsgemeinschaft (DFG) for financial support. They gave me the possibility to focus on my research, to meet experts by visiting conferences and to do part of my research in foreign countries.
I would especially like to thank Prof. Ivan Damgård and Prof. Ronald Cramer for supervising me during my time in Århus, for many helpful comments, always having time for me and to answer my questions, and for having the idea of considering the ECPS and branching programs. Additionally, I thank the European Union for their financial support.
Furthermore, I would like to thank Steven Galbraith for helpful discussions about the elliptic curve Paillier scheme. I thank Matthias Krause, Annegret Weng, Ralf Gerkmann and my colleagues at the BSI for fruitful discussions.
Besides that I thank my colleagues, John Malone-Lee, my boyfriend André and all the others for carefully reading my thesis and correcting my English.
Finally, but most important I thank my parents and all my good friends for being there for me. I owe special thanks to André for always staying at my side and always supporting me especially by coming with me to Denmark.

Thank you all!

Dörte Rappe

Contents

Chapter 1

Introduction

The demand for privacy of digital data and of more complex structures like algorithms has become stronger during the last few years. This goes hand in hand with the growth of communication networks like the Internet and the vastly growing number of electronic devices. On the one hand these devices enable a great variety of attacks on digital goods and on the other hand they are vulnerable to attacks such as the manipulation or destruction of data and the theft of sensitive information. For storing and reading data securely there exist several possibilities to guarantee privacy such as data encryption and tamper resistent hardware. The problem becomes more complex when asking for the possibility to compute (publicly) with private data or to modify functions or algorithms in such a way that they are still executable while their privacy is ensured. This is where homomorphic cryptosystems can be used since they enable computations with encrypted data.

In 1978 Rivest et al. [73] were the first to ask (implicitly) for homomorphic encryption schemes. Unfortunately their "privacy homomorphisms" were broken a couple of years later by Brickell and Yacobi [15].

The question rose again in 1991 when Feigenbaum and Merritt [38] asked: "*Is there an encryption function* $E()$ *such that both* $E(x + y)$ *and* $E(xy)$ *are easy to compute from* $E(x)$ *and* $E(y)$*?*" They were asking explicitly for so called algebraically homomorphic encryption schemes. Unfortunately, there has been little progress made in determining whether such encryption schemes exist that are efficient and secure, although it is one of the crucial open problems in cryptography. We cannot settle this question here but we prove that it is possible to obtain algebraically homomorphic cryptosystems given a homomorphic cryptosystem on a special non-abelian group. This may be viewed as a first step in answering the above question; However, in this thesis we do not consider this problem in more detail. Instead we mainly focus on applications of homomorphic cryptosystems.

Main Results

This thesis is organised into three main parts. In the first part we consider algebraically homomorphic schemes since they are of pivotal importance in designing powerful cryptographic protocols. However, as we already mentioned it is not clear yet whether such (efficient and secure) schemes even exist. The first contribution of this thesis is a new approach to the problem: constructing algebraically homomorphic cryptosystems from encryption schemes that are homomorphic on special non-abelian groups (see Section 2.1). Based on a construction of Ben-Or and Cleve [10] and a fact from projective geometry we observe that an algebraically homomorphic encryption scheme can be constructed from a homomorphic encryption scheme on the symmetric group on seven elements. Hence, the search for algebraically homomorphic schemes can be reduced to the search for homomorphic schemes on special non-abelian groups.
In the second part we analyse possible applications of algebraically homomorphic schemes. Owing to the lack of such schemes we design solutions based on homomorphic schemes only. Clearly, these cannot be as powerful as solutions based on algebraically homomorphic schemes. Nevertheless, we present as a second contribution a provably secure, non-interactive solution for encrypting functions given by polynomial branching programs and for encrypting data that is computed by such functions (see Chapter 3). This is an enlargement of the set of encryptable and still executable functions. We are now able to encrypt functions from the more general class of polynomial branching programs instead of functions represented by NC^1 circuits. Thus it is an improvement of one of the main applications of homomorphic cryptosystems. Furthermore, we introduce a new property that offers more possibilities concerning computations with encrypted data and encrypted functions,

respectively. We demonstrate examples of homomorphic schemes having this property that we call *key swapping*.
In the third part we describe an example of a homomorphic scheme that allows for realising the solutions presented in the previous parts. We generalise this scheme to obtain a wide variety of applications. More concretely, as a final contribution we develop a threshold decryption version of the elliptic curve Paillier scheme [42]. This version can be proven to be as secure as a centralised scheme with a trusted player who performs the decryption, i. e., it can be proven to be semantically secure against a static adversary in the random oracle model. We construct this threshold scheme in such a way that it is especially suited for several applications. Based on our new scheme we present various protocols that are the first elliptic curve versions of this kind. These protocols build the main tool for applying our scheme to different scenarios such as multiparty computation and voting schemes. Finally, by modifying the original elliptic curve Paillier scheme we are able to base a special mixed commitment scheme on it.

Outline of this Thesis

We begin with an introduction to homomorphic cryptosystems in Chapter 2 where we provide definitions and explain the relationship between homomorphic and algebraically homomorphic encryption schemes in Section 2.1, and briefly introduce some of the main applications in Section 2.2.

In Chapter 3 we present an improved solution to one of the main applications of homomorphic cryptosystems "computing with encrypted data" in Section 3.2 and closely related "computing with encrypted functions" in Section 3.3. After quickly introducing these issues we proceed with describing the concept of branching programs. We then present two provably secure and non-interactive protocols with zero-error that allow computation with encrypted data and functions, respectively. Finally, we define a new property of homomorphic schemes that we call key swapping. We demonstrate why this property is useful in the context of computations with encrypted data and functions. Furthermore, we give two examples of homomorphic schemes that support key swapping.

In Chapter 4 we present a scheme that has exactly the properties needed for our solution presented in Chapter 3: Galbraith's elliptic curve Paillier scheme [42]. We begin with summarising this semantically secure, probabilistic, additively homomorphic scheme and its generalisation by Galbraith. Thereafter we construct a further semantically secure generalisation in Section 4.4, namely a threshold decryption version as well as a length-flexible variant. This threshold cryptosystem is built in such a way that it is suitable for important applications. Subsequently, we design several auxiliary protocols in Section 4.5 to provide the basis for applications of our generalised elliptic curve Paillier scheme. These applications are considered in Sections 4.6, 4.7, and 4.8. They include a multiparty protocol, a protocol for electronic voting, and a commitment scheme for which we have to slightly modify the underlying elliptic curve Paillier scheme.

Finally, in Chapter 5 we conclude this work and give an outlook on further questions that are of interest.

Chapter 2

Homomorphic Cryptosystems

During the last few years homomorphic encryption schemes have been studied extensively since they have become more and more important in many different cryptographic protocols such as, e. g., voting protocols. In this chapter we introduce homomorphic cryptosystems in three steps ("what", "how", and "why") that reflect the main aspects. We start by defining homomorphic cryptosystems and algebraically homomorphic cryptosystems. Then we develop a method to construct algebraically homomorphic schemes given special homomorphic schemes. Finally, we describe applications of homomorphic schemes. A general and more detailed introduction to homomorphic cryptosystems can be found in [72].

Definition 2.0.1. Let the message space $(\mathcal{M}, \circ)$ be a finite (semi-)group, and let σ be the security parameter. A *homomorphic public-key encryption scheme* (or *homomorphic cryptosystem*) on $\mathcal{M}$ is a quadruple (K, E, D, A) of probabilistic, expected polynomial time algorithms, satisfying:

Key Generation: On input 1^σ the algorithm K outputs an encryption/decryption key pair $(k_e, k_d) = k \in \mathcal{K}$ where $\mathcal{K}$ denotes the key space.[1]

Encryption: On inputs 1^σ, k_e, and an element $m \in \mathcal{M}$ the encryption algorithm E outputs a ciphertext $c \in \mathcal{C}$ where $\mathcal{C}$ denotes the ciphertext space.

[1] Usually, we are interested in the running time of the algorithm K as a function of σ rather than $\log \sigma$, i. e., we want to allow expected polynomial time in the security parameter σ to generate a key. Therefore, technically we need to think of K as being given σ in unary notation. This is denoted by 1^σ.

Decryption: The decryption algorithm D is deterministic. On inputs 1^σ, k, and an element $c \in \mathcal{C}$ it outputs an element in the message space $\mathcal{M}$ so that for all $m \in \mathcal{M}$ it holds: if $c = E(1^\sigma, k_e, m)$ then $\text{Prob}[D(1^\sigma, k, c) \neq m]$ is negligible, i. e., it holds that $\text{Prob}[D(1^\sigma, k, c) \neq m] \leq 2^{-\sigma}$.

Homomorphic Property: A is an algorithm that on inputs $1^\sigma, k_e$, and elements $c_1, c_2 \in \mathcal{C}$ outputs an element $c_3 \in \mathcal{C}$ so that for all $m_1, m_2 \in \mathcal{M}$ it holds: if $m_3 = m_1 \circ m_2$ and $c_1 = E(1^\sigma, k_e, m_1), c_2 = E(1^\sigma, k_e, m_2)$ then

$$\text{Prob}[D(A(1^\sigma, k_e, c_1, c_2))] \neq m_3]$$

is negligible.

Informally speaking, a homomorphic cryptosystem is a cryptosystem with the additional property that there exists an efficient algorithm to compute an encryption of the sum or the product, of two messages given the public key and the encryptions of the messages but not the messages themselves.

If $\mathcal{M}$ is an additive (semi-)group then the scheme is called *additively homomorphic* and the algorithm A is called Add. Otherwise the scheme is called *multiplicatively homomorphic* and the algorithm A is called Mult.

Remark 2.0.2.

1. Note that for a homomorphic encryption scheme to be efficient it is crucial to make sure that the size of the ciphertexts remains polynomially bounded in the security parameter σ during repeated computations.

2. The security aspects, definitions, and models of homomorphic cryptosystems are the same as usually for cryptosystems.

If the encryption algorithm E gets as additional input a uniform random number r of a set $\mathcal{Z}$, the encryption scheme is called *probabilistic* otherwise it is called *deterministic*. Hence if a cryptosystem is probabilistic there belong several different ciphertexts to one message depending on the random number $r \in \mathcal{Z}$. But note that as before the decryption algorithm remains deterministic, i. e. there is just one message belonging to a given ciphertext. (See below Example 2.0.4.) Furthermore, in a probabilistic, homomorphic cryptosystem the algorithm A should be probabilistic, too to hide the input ciphertexts. For instance, this can be realised by applying a blinding algorithm (see Definition 2.0.7) on a (deterministic) computation of the encryption of the product and of the sum, respectively.

Notation 2.0.3. In the following we will omit the security parameter σ and the public key in the description of the algorithms. We will write $E_{k_e}(m)$ or $E(m)$ for $E(1^\sigma, k_e, m)$ and $D_k(c)$ or $D(c)$ for $D(1^\sigma, k, c)$ when no misunderstanding is possible. If the scheme is probabilistic we will also write $E_{k_e}(m)$ or $E(m)$ as well as $E_{k_e}(m, r)$ or $E(m, r)$ for $E(1^\sigma, k_e, m, r)$. Furthermore, we will write

$$\mathrm{A}(E(m), E(m')) = E(m \circ m')$$

to denote that the algorithm A (either Add or Mult) is applied on two encryptions of the messages $m, m' \in (\mathcal{M}, \circ)$ and outputs an encryption of $m \circ m'$, i. e., it holds that

$$D(\mathrm{A}(1^\sigma, k_e, E_{k_e}(m), E_{k_e}(m'))) = m \circ m'$$

except with negligible probability.

Example 2.0.4. Here we give an example of a deterministic, multiplicatively homomorphic scheme and an example for a probabilistic, additively homomorphic scheme.

1. The classical RSA scheme [74] is an example of a deterministic, multiplicatively homomorphic cryptosystem on $\mathcal{M} = (\mathbb{Z}/N\mathbb{Z}, \cdot)$, where N is the product of two large primes. As ciphertext space we have $\mathcal{C} = (\mathbb{Z}/N\mathbb{Z}, \cdot)$ and as key space we have $\mathcal{K} = \{(k_e, k_d) = ((N, e), d) \mid N = pq, ed \equiv 1 \bmod \varphi(N)\}$. The encryption of a message $m \in \mathcal{M}$ is defined as $E_{k_e}(m) := m^e \bmod N$ and for decryption of a ciphertext $E_{k_e}(m) = c \in \mathcal{C}$ we compute $D_{k_e,k_d}(c) := c^d \bmod N = m \bmod N$. Obviously, the encryption of the product of two messages can be efficiently

computed by multiplying the corresponding ciphertexts, i. e.,

$$\begin{aligned} E_{k_e}(m_1 \cdot m_2) &= (m_1 m_2)^e \bmod N \\ &= ({m_1}^e \bmod N)({m_2}^e \bmod N) \\ &= E_{k_e}(m_1) \cdot E_{k_e}(m_2) \end{aligned}$$

where $m_1, m_2 \in \mathcal{M}$. Hence, the algorithm Mult can easily be implemented as

$$\mathrm{Mult}(E_{k_e}(m_1), E_{k_e}(m_2)) := E_{k_e}(m_1) \cdot E_{k_e}(m_2).$$

Usually in the RSA scheme as well as in most schemes based on the difficulty of factoring the security parameter σ is the bit length of N. For instance, $\sigma = 1024$ is a common security parameter.

2. The Goldwasser-Micali scheme, proposed in [45] is an example of a probabilistic, additively homomorphic cryptosystem on $\mathcal{M} = (\mathbb{Z}/2\mathbb{Z}, +)$ with $\mathcal{C} = \mathcal{Z} = (\mathbb{Z}/N\mathbb{Z})^*$ where $N = pq$ is the product of two large primes. We have

$$\mathcal{K} = \{(k_e, k_d) = ((N, a), (p, q)) \mid N = pq, a \in (\mathbb{Z}/N\mathbb{Z})^* : \left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1\}.$$

Since this scheme is probabilistic, the encryption algorithm gets as additional input a random value $r \in \mathcal{Z}$. We define $E_{k_e}(m, r) := a^m r^2 \bmod N$ and

$$D_{(k_e,k_d)}(c) := \begin{cases} 0, & \text{if } c \text{ is a square} \\ 1, & \text{else} \end{cases}$$

It holds that

$$E_{k_e}(m_1, r_1) \cdot E_{k_e}(m_2, r_2) = E_{k_e}(m_1 + m_2, r_1 r_2).$$

Thus the algorithm Add can be efficiently implemented e. g. as

$$\begin{aligned} \mathrm{Add}(E_{k_e}(m_1, r_1), E_{k_e}(m_2, r_2), r_3) &= E_{k_e}(m_1, r_1) \cdot E_{k_e}(m_2, r_2) \cdot \underbrace{{r_3}^2 \bmod N}_{E_{k_e}(0, r_3)} \\ &= E_{k_e}(m_1 + m_2, r_1 r_2 r_3) \end{aligned}$$

where $m_1, m_2 \in \mathcal{M}$ and $r_1, r_2, r_3 \in \mathcal{Z}$. Note that as ready mentioned this algorithm should be probabilistic, i. e., it obtains a random number r_3 as additional input.

A public-key homomorphic encryption scheme on a (semi-)ring $(\mathcal{M}, +, \cdot)$ can be defined in an analogous manner. Such schemes consist of two algorithms Add and Mult for the homomorphic property instead of one algorithm A, i. e., it is additively and multiplicatively homomorphic at the same time. Such schemes are called *algebraically homomorphic*.

Definition 2.0.5. An additively homomorphic encryption scheme on a (semi-)ring $(\mathcal{M}, +, \cdot)$ is called *scalar homomorphic* if there exists a probabilistic, expected polynomial time algorithm Mixed-Mult that on inputs $1^\sigma, k_e,\ s \in \mathcal{M}$ and an element $c \in \mathcal{C}$ outputs an element $c' \in \mathcal{C}$ so that for all $m \in \mathcal{M}$ it holds that: if $m' = s \cdot m$ and $c = E(1^\sigma, k_e, m)$ then $\text{Prob}[D(\text{Mixed-Mult}(1^\sigma, k_e, s, c)) \neq m']$ is negligible.

Thus in a scalar homomorphic scheme it is possible to compute an encryption $E(1^\sigma, k_e, s \cdot m) = E(1^\sigma, k_e, m')$ of a product of two messages $s, m \in \mathcal{M}$ given the public key k_e and an encryption $c = E(1^\sigma, k_e, m)$ of one message m and the other message s as a plaintext.

Obviously any scheme that is algebraically homomorphic is scalar homomorphic, too.

Notation 2.0.6. We will denote by

$$\text{Mixed-Mult}(m, E(m')) = E(mm')$$

if

$$D(\text{Mixed-Mult}(1^\sigma, k_e, m, E_{k_e}(m')) = m \cdot m'$$

holds except with negligible probability.

Definition 2.0.7. A *blinding algorithm* is a probabilistic, polynomial time algorithm, which on inputs $1^\sigma, k_e$, and $c \in E_{k_e}(m, r)$ where $r \in \mathcal{Z}$ is randomly chosen outputs another encryption $c' \in E_{k_e}(m, r')$ of m where $r' \in \mathcal{Z}$ is chosen uniformly at random.

For instance, in a probabilistic, homomorphic cryptosystem on $(\mathcal{M}, \circ)$ the blinding algorithm can be realised by applying the algorithm A on the ciphertext c and an encryption of the identity element in $\mathcal{M}$.

Remark 2.0.8. If $\mathcal{M}$ is isomorphic to $\mathbb{Z}/n\mathbb{Z}$ if $\mathcal{M}$ is finite, or to $\mathbb{Z}$ otherwise, then the algorithm Mixed-Mult can easily be implemented using a double and add algorithm. This is combined with a blinding algorithm if the scheme is probabilistic [24]. Hence, every additively homomorphic cryptosystem on $\mathbb{Z}/n\mathbb{Z}$ or $\mathbb{Z}$ is also scalar homomorphic and the algorithm Mixed-Mult can be efficiently implemented (see also [75]).

Remark 2.0.9. Since most of the existing additively homomorphic cryptosystems (e. g., [44, 7, 62, 65, 67, 77, 31, 42]) are defined on $\mathbb{Z}/n\mathbb{Z}$ for some $n \in \mathbb{N}$ the algorithm Mixed-Mult can in general be implemented efficiently.[2]

2.1 Algebraically Homomorphic Cryptosystems

As already mentioned the existence of an efficient and secure algebraically homomorphic cryptosystem has been a long standing open question. In this section we first present related work considering this problem. Thereafter we describe the relationship between algebraically homomorphic schemes and homomorphic schemes on special non-abelian groups. More precisely, we prove that a homomorphic encryption scheme on the non-abelian group $(S_7, \cdot)$, the symmetric group on seven elements, allows to construct an algebraically homomorphic encryption scheme on $(\mathbb{F}_2, +, \cdot)$. An algebraically homomorphic encryption scheme on $(\mathbb{F}_2, +, \cdot)$ can also be obtained from a homomorphic encryption scheme on the special linear group $(SL(3, 2), \cdot)$ over $\mathbb{F}_2$. Furthermore, using coding theory an algebraically homomorphic encryption on an arbitrary finite ring or field could be obtained given a homomorphic encryption scheme on one of these non-abelian groups. These observations could be a first step to solve the problem whether efficient and secure algebraically homomorphic schemes exist. Many authors have tried to solve this problem. In 1996, Boneh and Lipton have proven that under a reasonable assumption every deterministic, algebraically

[2] We denote by $\mathbb{N}$ the set of positive integers.

homomorphic cryptosystem can be broken in sub-exponential time [13]. This may be seen as a negativ result concerning the existence although most existing cryptosystem, e. g., the RSA scheme (see Example 2.0.4) or the ElGamal scheme (see Section 3.4.1), can be broken in sub-exponential time, too. Furthermore, if we are seeking for algebraically homomorphic public-key schemes on small fields or rings such as $\mathcal{M} = \mathbb{F}_2$, obviously such a scheme has to be probabilistic to be secure.
Other authors have tried to find candidates for algebraically homomorphic schemes. In 1993, Fellows and Koblitz presented an algebraic public-key cryptosystem called Polly Cracker [39]. It is algebraically homomorphic and provably secure. Unfortunately, the scheme has a number of difficulties and is not efficient concerning the ciphertext length. Firstly, Polly Cracker is a polynomial-based system. Therefore, computing an encryption of the product $E(m_1 \cdot m_2)$ of two messages m_1 and m_2 by multiplying the corresponding ciphertext polynomials $E(m_1)$ and $E(m_2)$, leads to an exponential blowup in the number of monomials. Hence, during repeated computations there is an exponential blowup in the ciphertext length.
Secondly, all existing instantiations of Polly Cracker suffer from further drawbacks (see e. g., [59]). They are either insecure since they succumb to certain attacks, they are too inefficient to be practical, or they loose the algebraically homomorphic property. Hence it is far from clear how such kind of schemes could be turned into efficient and secure algebraically homomorphic encryption schemes. A detailed analysis and description of these schemes can be found in [61].

In 2002, J. Domingo-Ferrer [36] developed a probabilistic, algebraically homomorphic secret-key cryptosystem. But as before this scheme is not efficient since there is an exponential blowup in the ciphertext length during repeated multiplications. Furthermore, it was recently broken by Wagner and Bao [81, 3].

Thus considering homomorphic encryption schemes on groups instead of rings seems more promising to solve the presented question. It brings us closer to structures that have been successfully used in cryptography. The following theorem shows that indeed the search for algebraically homomorphic schemes can be reduced to the search for homomorphic schemes on special non-abelian groups.

Theorem 2.1.1. *The following statements are equivalent:*

1. *There exists an algebraically homomorphic encryption scheme on* $(\mathbb{F}_2, +, \cdot)$.

2. *There exists a homomorphic encryption scheme on the symmetric group* $(S_7, \cdot)$.

The following proof is based on an idea of Tomas Sander.

Proof. $1 \Rightarrow 2$ This direction of the proof follows immediately and it holds for an arbitrary finite group since operations of finite groups can always be implemented by Boolean circuits. Let S_7 be represented as a subset of $\{0,1\}^l$, where e.g. $l = 21$ can be chosen[3], and let C be a circuit with addition and multiplication gates that takes as inputs the binary representations of elements $m_1, m_2 \in S_7$ and outputs the binary representation of $m_1 m_2$. If we have an algebraically homomorphic encryption scheme $(K, E, D, \mathrm{Add}, \mathrm{Mult})$ on $(\mathbb{F}_2, +, \cdot)$ then we can define a homomorphic scheme $(\widetilde{K}, \widetilde{E}, \widetilde{D}, \widetilde{\mathrm{Mult}})$ on S_7 by defining $\widetilde{E}(m) = (E(s_0), \ldots, E(s_{l-1}))$ where $(s_0, \ldots, s_{l-1})$ denotes the binary representation of m. $\widetilde{\mathrm{Mult}}$ is constructed by substituting the addition gates in C by Add and the multiplication gates by Mult. $\widetilde{K}$ and $\widetilde{D}$ are defined in the obvious way.

$2 \Rightarrow 1$ The proof has two steps. First we use a construction of Ben-Or and Cleve [10] to show that the field $(\mathbb{F}_2, +, \cdot)$ can be encoded in the special linear group $(SL(3,2), \cdot)$ over $\mathbb{F}_2$. Then we apply a theorem from projective geometry to show that $(SL(3,2), \cdot)$ is a subgroup of S_7. This proves the claim.

We map

$$\begin{array}{ccc} (\mathbb{F}_2, +, \cdot) & \overset{encoding}{\rightarrow} & SL(3,2) \\ x & \mapsto & \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} =: M(x). \end{array}$$

It is easy to see that:

1. $M(x)M(y) = M(x+y)$
2. $\exists\, T \in SL(3,2)$:

$$TM(x)T^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix} \text{ and}$$

[3] Since $\log_2(7!) \approx 12.3$, the minimal possible number of bits is $l = 13$.

$\exists\ S \in SL(3,2):$

$$SM(y)S^{-1} = \begin{pmatrix} 1 & y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}:$$

$$M(xy) = [\begin{pmatrix} 1 & y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix}],$$

where $[A, B]$ denotes the commutator of the matrices A and B.

Therefore the operations $+$ and $\cdot$ in $\mathbb{F}_2$ can be performed on $M(0)$ and $M(1)$ using the constants S and T, and the multiplication in the group $SL(3,2)$. Given a homomorphic encryption scheme (K, E, D, Mult) on $(SL(3,2), \cdot)$ we can thus easily construct an algebraically homomorphic scheme on $(\mathbb{F}_2, +, \cdot)$ using $E(S), E(T)$, and the algorithm Mult.
We now use the fact that $SL(3,2)$ is the group of collineations on the projective plane over $\mathbb{F}_2$ (see e. g. [52]). As this plane has 7 points, $SL(3,2)$ can be embedded in S_7, which concludes the proof. □

Remark 2.1.2. An analysis of the proof of Theorem 2.1.1 shows that using the described construction the matrices S, T, and $M(x)$ are necessary for the encoding. We have

$$\langle M(1), S, T\rangle = SL(3,2).$$

Hence, there is no proper subgroup of $SL(3,2)$ that can be used to encode $\mathbb{F}_2$ in the above way. We additionally note that S_7 is the smallest symmetric group in which $SL(3,2)$ can be embedded since $SL(3,2)$ contains an element of order 7. For instance, the following element has order 7:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

An interesting question is whether there are "smaller" or "simpler" groups than S_7

and $SL(3,2)$ that allow to encode the field $\mathbb{F}_2$.

From the proof of Theorem 2.1.1 we obtain directly the following corollary.

Corollary 2.1.3. The following statements are equivalent:

1. There exists an algebraically homomorphic encryption scheme on $(\mathbb{F}_2, +, \cdot)$.

2. There exists a homomorphic encryption scheme on the special linear group $(SL(3,2), \cdot)$.

Remark 2.1.4. In exactly the same way as in the proof of Theorem 2.1.1 it is possible to encode the finite field $(\mathbb{F}_p, +, \cdot)$ in the special linear group $(SL(3,p), \cdot)$.

Homomorphic encryption schemes on groups have been extensively studied. For instance, we have homomorphic schemes on groups $(\mathbb{Z}/_{M\mathbb{Z}}, +)$, for M being a smooth number (e. g., [45, 8, 62]), for $M = pq$ being an RSA modulus [67, 42], and for groups $((\mathbb{Z}/_{N\mathbb{Z}})^*, \cdot)$ where N is an RSA modulus. All known efficient and secure schemes today are homomorphic on abelian groups, however S_7 and $SL(3,2)$ are non-abelian. Sander, Young and Yung [77] had asked explicitly whether there is a homomorphic encryption scheme on non-abelian groups. The reduction above motivates that this is indeed an important question and that non-abelian groups should be studied from this perspective.
Unfortunately, we can not describe an efficient homomorphic encryption scheme on a non-abelian group here either. Although non-abelian groups had been previously used to construct encryption schemes ([58, 66, 82, 49]) the resulting schemes were not homomorphic in the sense that we need for computing efficiently on encrypted data (see Chapter 3).

In [49] Grigoriev and Ponomarenko propose a new definition of homomorphic cryptosystems on which they base a method to construct homomorphic cryptosystems over arbitrary finite groups including non-abelian groups. Their construction method is based on the fact that every finite group is an epimorphic image of a free product of finite cyclic groups. It uses existing homomorphic encryption schemes on finite cyclic groups as building blocks to obtain homomorphic encryption schemes on arbitrary finite groups. Since the ciphertext space of the so obtained encryption scheme is a free product of groups an exponential blowup of the ciphertext lengths during repeated computations follows. The reason is that the length of the product of two elements x and y of a free product is in general the sum of the length of x and the length of y. Hence their technique suffers from the same drawback as before and thus does not lead to an efficient cryptosystem. Note that using this construction it is possible to construct a homomorphic encryption scheme on the symmetric group S_7 and on the special linear group $SL(3,2)$. If we combine this with Theorem 2.1.1 we can construct an algebraically homomorphic cryptosystem on the finite field $(\mathbb{F}_2, +, \cdot)$. Unfortunately, the exponential blowup owing to the construction method in the homomorphic encryption scheme on S_7 and on $SL(3,2)$, respectively, would lead to an exponential blowup in $\mathbb{F}_2$ and hence leaves the question open if an efficient algebraically homomorphic cryptosystem on $\mathbb{F}_2$ exists.

In [50] Grigoriev and Ponomarenko propose another method to encrypt arbitrary finite groups homomorphically. This method is based on the difficulty of the membership problem for groups of integer matrices, while in [49] it is based on the difficulty of factoring. However, as before this scheme is not efficient. Furthermore, in [50] an algebraically homomorphic cryptosystem over finite commutative rings is proposed, but owing to its immense size it is infeasible to implement.

2.2 Applications and Properties of Homomorphic Cryptosystems

An inherent drawback of homomorphic cryptosystems is that attacks might exploit their additional structure. For instance, using plain RSA [74] for signing, the multiplication of two signatures yields a valid signature of the product of the two corresponding messages. Although there are many ways to avoid such attacks for instance by the application of hash functions, the use of redundancy or probabilistic schemes, this potential weakness leads us to the question why homomorphic schemes are used in some situations instead of conventional cryptosystems. The main reason for the interest in homomorphic cryptosystems is its wide application scope. There are theoretical as well as practical applications in different areas of cryptography.

In the following we list some of the main applications and properties of homomorphic schemes and summarise the idea behind them. We will describe some of these applications in more detail throughout the following chapters.

- Protection of mobile agents:
  One of the most interesting and demanding application of homomorphic cryptosystems is the protection of mobile agents. As it was shown in Theorem 2.1.1 a homomorphic encryption scheme on a special non-abelian group would lead to an algebraically homomorphic cryptosystem on the finite field $\mathbb{F}_2$. Since all conventional computer architectures are based on binary strings and only require multiplication and addition, such homomorphic cryptosystems would offer the possibility to encrypt a whole program so that it is still executable. Hence, it could be used to protect mobile agents against malicious hosts by encrypting them. More details about this idea can be found in [76].

  - Computing with encrypted functions:
    This is a special case of the protection of mobile agents. In such scenarios a secret function is publicly evaluated in such a way that the function remains secret. Using homomorphic cryptosystems the encrypted function can be evaluated which guarantees its privacy. This will be further discussed in the next chapter.

  - Computing with encrypted data:
    Homomorphic schemes offer the possibility to compute publicly with secret data such that it remains secret. This can be done by encrypting the data in advance and then exploiting the homomorphic property to compute with encrypted data. The area "computing with encrypted data" can be shown to be equivalent to the case of "computing with encrypted functions". The idea behind it will be further explained in the following chapter.

- Multiparty computation:
  In multiparty computation schemes several parties want to compute a common, public function on their inputs while keeping their individual inputs private. This belongs to the area of "computing with encrypted data". Usually

in multiparty computation protocols we have a set of $n \geq 2$ players whereas in computing with encrypted data scenarios $n = 2$. Furthermore, in multiparty computation protocols the function that should be computed is publicly known, whereas in the area of "computing with encrypted data" it is a private input of one party. A protocol for multiparty computations will be given in Section 4.6.

- Secret sharing schemes:
  In a secret sharing scheme several parties share a secret so that no party can reconstruct the secret by itself, but if many parties cooperate they are able to reconstruct it. Here, the homomorphic property implies that the composition of the secrets' shares are shares of the secrets' composition.

- Threshold schemes:
  Secret sharing schemes and multiparty computation schemes are examples of threshold schemes that are based on the homomorphic property. An example of a threshold decryption scheme is given in Chapter 4.

- Zero-knowledge proofs:
  This is a fundamental primitive of cryptographic protocols and thus an example of a theoretical application of homomorphic cryptosystems. Zero-knowledge proofs are used to prove knowledge of some private information. For instance, consider the case where a user has to prove his identity to a host by logging in with his account and private password. Obviously, in such a protocol the user wants his private information (i. e., his password) to stay private and not to be leaked during the protocol. Zero-knowledge proofs guarantee that the protocol communicates exactly the knowledge that was intended, and no (zero) extra knowledge. An example of an honest-verifier zero-knowledge proof that uses the homomorphic property is given in Section 4.5.3. For further examples of zero-knowledge proofs using the homomorphic property see [22].

- Election schemes:
  In election schemes the homomorphic property provides a tool to obtain the tally given the encrypted votes without decrypting the individual votes. In Section 4.7 we will explain this further.

- Watermarking and fingerprinting schemes:
  Digital watermarking and fingerprinting schemes embed additional information into digital data. The homomorphic property is used to add a mark to previously encrypted data. In general watermarks are used to identify the owner/seller of digital goods to ensure the copyright. In fingerprinting schemes the person who buys the data should be identifiable by the merchant to ensure that data is not illegally redistributed. Further properties of such schemes can be found in [70] or [2].

- Oblivious transfer:
  Oblivious transfer is another cryptographic primitive. Usually in a two-party 1-out-of-2 oblivious transfer protocol the first party sends a bit to the second party in such a way that the second party receives it with probability 1/2, without the first party knowing whether or not the second party received the bit. An example of such a protocol that uses the homomorphic property can be found in [60].

- Commitment schemes:
  Commitment schemes are another fundamental primitive of cryptographic protocol theory. In a commitment scheme a player makes a commitment. He is able to choose a value from some set and commit to his choice such that he can no longer change his mind. He does not have to reveal his choice although he may do so at some later point. Commitment schemes that use the homomorphic property will be introduced in Section 4.8.

- Lottery protocols:
  Usually in a cryptographic lottery a number pointing to the winning ticket has to be jointly and randomly chosen by all participants. Using a homomorphic encryption scheme this can be realized as follows: Each player chooses a random number which he encrypts. Then using the homomorphic property the encryption of the sum of the random values can be efficiently computed. The combination of this and a threshold decryption scheme (like e. g. proposed in Section 4.4) leads to the desired functionality. See [41] for further details.

- Mix-nets:
  Mix-nets are protocols that provide anonymity for senders by collecting encrypted messages from several users. For instance, one can consider mix-nets that collect ciphertexts and output the corresponding plaintexts in a randomly permuted order. In such a scenario privacy is obtained by requiring that the permutation, matching inputs to outputs, is secret to anyone except the mix-net. In particular, determining a correct input/output pair, i. e., a ciphertext with corresponding plaintext, should not be more effective then guessing one at random. A desirable property to build such mix-nets is re-encryption. As will be mentioned below, this is provided by the use of homomorphic cryptosystems as building block. For instance, see [48] and [31] for details.

Examples of useful properties of homomorphic schemes are the following:

- Re-randomizable encryption/re-encryption:
  Re-randomizable cryptosystems (see e. g. [51]) are probabilistic cryptosystems with the additional property that given the public key k_e and an encryption $E_{k_e}(m, r)$ of a message $m \in \mathcal{M}$ under the public key k_e and a random number $r \in \mathcal{Z}$ it is possible to efficiently convert $E_{k_e}(m, r)$ into another encryption $E_{k_e}(m, r')$ that is perfectly indistinguishable from a "fresh" encryption of m under the public key k_e. This property is also called re-encryption.
  Obviously every probabilistic homomorphic cryptosystem is re-randomizable: Without loss of generality we assume that the cryptosystem is additively homomorphic. Given $E_{k_e}(m, r)$ and the public key k_e we can compute $E_{k_e}(0, r'')$ for a random number r'' and hence compute

  $$\mathrm{Add}(E_{k_e}(m, r), E_{k_e}(0, r'')) = E_{k_e}(m + 0, r') = E_{k_e}(m, r'),$$

  with r' being an appropriate random number. Note that this is exactly what a blinding algorithm (see Definition 2.0.7) does.

- Random self-reducibility:
  Along with the possibility of re-encryption comes the property of random self-reducibility concerning the problem of computing the plaintext from the

ciphertext. A cryptosystem is called random self-reducible if any algorithm that can break a non-trivial fraction of ciphertexts can also break a random instance with significant probability. This property is further discussed in e. g. [32] or [77].

- Verifiable encryptions/fair encryptions:
  If an encryption is verifiable it provides a mechanism to check the correctness of encrypted data without compromising secrecy. For instance, this is useful in voting schemes to convince any observer that the encrypted name of a candidate, i. e., the encrypted vote is indeed in the list of candidates. A cryptosystem with this property that is based on homomorphic encryptions can be found in [71] (see also Section 4.7). Note that in the literature verifiable encryptions are also called fair encryptions.

Chapter 3

Computing with Encrypted Data and Encrypted Functions

The search for an efficient algebraically homomorphic cryptosystem is a long standing open problem. A major motivation is the protection of mobile agents and the secure computation with encrypted functions or encrypted data as already mentioned in Section 2.2. Until now no efficient algebraically homomorphic scheme has been found. All candidates for those schemes suffer from the same drawback (see Section 2.1): they have to deal with an exponential blowup in the ciphertext length during repeated multiplication. This is the main reason why their application scope is restricted to functions that can be implemented by NC^1 circuits, i. e., circuits of logarithmic depth. Thus other methods are needed to enable secure computations. We come up with a new approach for this problem. Instead of developing a new scheme or considering circuits we take a closer look at branching programs as a computational model for functions. Using existing efficient additively homomorphic cryptosystems like Paillier's cryptosystem [67] provides a tool to encrypt branching programs in order to enable secure computations. Thus in this chapter we suggest a provably secure and non-interactive method to compute with encrypted data and encrypted functions that are given by polynomial branching programs. This is the first solution using branching programs we are aware of. Furthermore, it is an enlargement of the class of functions that are encryptable efficiently and non-interactively, in such a way that they are still executable, from NC^1 to polynomial branching programs [4]. Moreover, we define a new property of homomorphic cryptosystems that we call key swapping. This property offers new possibilities in "computing with encrypted data" and "computing with encrypted functions" scenarios. Finally, we present two examples of homomorphic schemes supporting key swapping.

3.1 Branching Programs

In this section we introduce the basic concepts of branching programs that we will need in the following. The definition we present was recently introduced by Cramer et al. in [26]. It refers to branching programs over arbitrary rings. It includes the notion usually given in literature, e. g., in [83] as a special case. Therefore the new definition is more powerful which induces that known facts still hold for the new definition and that bounds might be improved.

Definition 3.1.1. A *branching program* of size λ on inputs $x_1, \ldots, x_n \in R^n$, where R is an arbitrary ring, is a quadruple $BP = (G, w, s, t)$ where $G = (V, E)$ is a directed acyclic graph with V being the set of vertices and E being the set of edges, w is a labelling function and s and t are two special vertices of the graph.
We may assume that $V = \{s = 1, \ldots, t = \lambda\}$ and that for each edge $(i, j) \in E$ it holds that $i < j$. Let w assign to each edge a degree-1 polynomial over R in the input variables, i. e.,

$$w(i,j) = \sum_{k=1}^{n} a_{lk} x_k + b_l, \;\; a_{lk}, b_l \in R \text{ and } 1 \leq l \leq \lambda^2.$$

For each directed path $\phi = (i_1, i_2, \ldots, i_k)$ from i_1 to i_k in G, the *weight* of ϕ is defined to be the product $w(i_1, i_2) \cdot w(i_2, i_3) \cdots w(i_{k-1}, i_k)$. $W(i, j)$ denotes the *total weight* of all directed paths from i to j for $i < j$ (viewed as a function of $x = (x_1, \ldots, x_n)$). That is

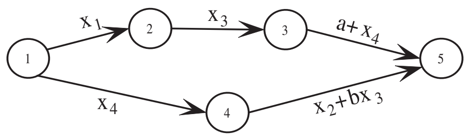
$$W(i,j) = \sum_{\phi(i,j)} \prod_{m=1}^{k-1} w(i_m, i_{m+1})$$

where $i = i_1$ and $j = i_k$ and $\phi(i, j)$ denotes a path from i to j.

The function $f : R^n \to R$ computed by a branching program BP is defined by

$$f(x) = W(1, \lambda)(x) = W(s, t)(x).$$

Example 3.1.2. Let R be a commutative ring with 1. As an example of a branching program of size 5 on inputs x_1, x_2, x_3, x_4 consider the following graph, where $x_1, x_2, x_3, x_4, a, b \in R$:



We have

$$w(1,2) = x_1,\ w(2,3) = x_3,\ w(3,5) = a + x_4,\ w(1,4) = x_4,\ w(4,5) = x_2 + bx_3.$$

Hence, the weight of the path $\phi = (1, 4, 5)$ is defined to be

$$w(1,4)w(4,5) = x_4 \cdot (x_2 + bx_3).$$

The total weight $W(i, j)$ equals $x_1x_3(a + x_4) + x_4(x_2 + bx_3)$ and thus the function computed by BP is

$$f(x) = x_1x_3(a + x_4) + x_4(x_2 + bx_3).$$

If R is a commutative ring with 1, the computation of $f(x)$ can be mapped to the computation of the determinant of a matrix in the following way (see e. g. [53]):

Let $x = (x_1, \ldots, x_n) \in R^n$. There is an adjacency matrix $H(x)$ belonging to each branching program $BP = (G, w, s, t)$ where $G = (V, E)$. If we assume that $V = \{s = 1, \ldots, t = \lambda\}$ and that for each edge $(i, j) \in E$ it holds that $i < j$, then $H(x)$ is an $\lambda \times \lambda$ matrix where entry (i, j) equals 0 if $(i, j) \notin E$ and $w(i, j)$ otherwise. Since G is a directed acyclic graph $H(x)$ is an upper triangular matrix which is nilpotent. If r is the number of edges of the longest path from $s = 1$ to $t = \lambda$ then $H(x)^{r+1} = 0$. When no misunderstanding is possible we will write H instead of $H(x)$.

Let *k-path* denote a path that consists of exactly k edges. The entry (i, j) of the power H^k equals the total weight of all directed k-paths from i to j. Hence,

$$H^* := \sum_{k=0}^{\infty} H^k = \sum_{k=0}^{r} H^k$$

is the matrix whose (i,j) entry equals $W(i,j)$. Since $H^{r+1} = 0$ we obtain that

$$(I - H)H^* = H^*(I - H) = \sum_{k=0}^{r} H^k - \sum_{k=0}^{r} H^{k+1} = H^0 = I$$

$$\Rightarrow H^* = (I - H)^{-1}.$$

Furthermore, the upper right entry $(1, \lambda) = (s, t)$ of H^*, denoted by $H^*_{1,\lambda}$, equals $W(1, \lambda)(x)$. This determines the output $f(x)$ of the function $f : R^n \to R$ computed by BP. To compute the entry $(1, \lambda)$ of H^* we can apply Cramer's rule. Hence

$$H^*_{1,\lambda} = W(1, \lambda) = ((I - H)^{-1})_{1,\lambda} = (-1)^{\lambda+1} \frac{\det M}{\det(I - H)},$$

where M denotes the submatrix of $I - H$ obtained by deleting the first column and the last (λ-th) row. Since H is an upper triangular matrix we have $\det(I - H)$=1. Hence

$$H^*_{1,\lambda} = (-1)^{\lambda+1} \det M = f(x).$$

Remark 3.1.3. If f is a Boolean function then this determinant equals either 0 or 1, i. e., to compute the function value it is sufficient to decide if the matrix M is of full rank.

Example 3.1.4. The adjacency matrix of the graph of Example 3.1.2 looks as follows:

$$H = \begin{pmatrix} 0 & x_1 & 0 & x_4 & 0 \\ 0 & 0 & x_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & a + x_4 \\ 0 & 0 & 0 & 0 & x_2 + bx_3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow H^2 = \begin{pmatrix} 0 & 0 & x_1x_3 & 0 & x_4(x_2+bx_3) \\ 0 & 0 & 0 & 0 & x_3(a+x_4) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$H^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & x_1x_3(a+x_4) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, H^4 = 0.$$

Since $H^* := \sum H^k$, it follows that

$$H^*_{1,5} = W(1,5)(x) = x_1x_3(a+x_4) + x_4(x_2+bx_3) = f(x).$$

The reason to look closer at branching programs is that every function f can be computed by a branching program, and many "natural" function families (in particular regular languages) can be computed by linear-size branching programs. Furthermore, the branching program size is no larger than the corresponding formula size of f [26]. However, polynomial-size branching programs are probably not powerful enough to efficiently compute all polynomial-time computable functions. This is owing to the fact that the power of polynomial-size branching programs coincides with that of different variants of log-space computation [54].

Remark 3.1.5. The upper bound for the size of branching programs computing Boolean functions $f : \{0,1\}^n \to \{0,1\}$ is exponential. For special classes of functions a smaller bound can be proven. An example of functions that can be computed by linear-size branching programs are Boolean functions $f(x) = 1 + \sum_{J \subseteq \{1,\ldots,n\}} \prod_{i \in J} x_i$. Hence, $f(x) = \prod_{i=1}^n (1 + x_i)$ and there exists a branching program of size $n+1$ computing f.

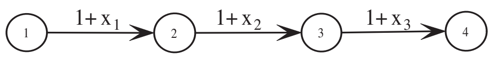
Example 3.1.6. Let

$$f(x_1,x_2,x_3) = 1 + \sum_{J \subseteq \{1,2,3\}} \prod_{i \in J} x_i = 1 + x_1 + x_2 + x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_2x_3,$$

where $x_i \in \{0,1\}$ for $1 \leq i \leq 3$. Then

$$\begin{aligned} f(x_1,x_2,x_3) &= 1+x_2+x_3+x_2x_3+x_1(1+x_2+x_3+x_2x_3) \\ &= (1+x_2+x_3+x_2x_3)(1+x_1) \\ &= (1+x_3+x_2(1+x_3))(1+x_1) = ((1+x_3)(1+x_2))(1+x_1). \end{aligned}$$

The following branching program of size 4 computes f:



Remark 3.1.7. An interesting application of branching programs - namely the construction of so-called randomizing polynomials - was introduced in [54, 55] and further generalised in [26]. These randomizing polynomials are a representation of functions by low-degree polynomials which is especially useful for the design of secure multiparty computation protocols.

3.2 Computing with Encrypted Data

In this section we consider an important application of homomorphic cryptosystems: *computing with encrypted data*. We first give an introduction to this problem by describing it and giving examples. Then we develop a protocol that solves the underlying problem. Our protocol is non-interactive, efficient for polynomial-size branching programs, and provably secure. It is an improvement to known solutions in the literature.

Consider the following Scenario 1. It was introduced as *secure circuit evaluation* in [1] (see also [85, 46]) and it can be found in the literature with several different notions, e. g., in [77, 16]. Note that in Section 3.3 we consider a scenario known as *computing with encrypted functions*, that is equivalent to Scenario 1.

Scenario 1:
Alice has a function $f : R^n \to R$ that she does not want to reveal and some secret inputs $x_1, \ldots, x_k \in R^k$. Bob has some secret inputs $y_1, \ldots, y_m \in R^m$ where $k + m = n$. Alice is willing to compute $f(x_1, \ldots, x_k, y_1, \ldots, y_m) =: f(x, y)$ on her

inputs $x_i, 1 \leq i \leq k$ and Bob's secret inputs $y_j, 1 \leq j \leq m$ so that only Bob learns the output of the function. Bob and Alice should learn nothing about the other ones input values besides the information that the output reveals.

Variants:

1. For some applications we could think about f being a public function (see e. g. [85, 46]). The solution given below can also be used to solve this scenario.

2. In some cases Alice should learn the output, too.

3. Only Alice should learn the output $f(x)$.

Remark 3.2.1. Obviously Alice's inputs can also be seen as a fixed part of her function f. Hence $k = 0$ is also possible.

Example 3.2.2. Now we give examples of the above scenario and its variants.

1. As an example of Scenario 1 we could think about Alice having an efficient algorithm for factoring that she uses to earn money with and thus wants to keep private. Bob has an input that he wants to have factorized. Alice offers Bob to factor his input for money.

2. As a typical example of the first variant we could think about Alice wanting to sell her computing power to Bob for a public algorithm.

3. If f is public - as in Variant 1 - and both Alice and Bob should learn the output - as in Variant 2 - we have the scenario usually considered in 2-party computation protocols. As an example we consider f as being the function that computes the maximum of two inputs. Alice and Bob now use f in order to decide who of them earns more money. Multiparty computation solutions for such a scenario differ from our solution since in the former one the roles of Alice and Bob are usually symmetric, i. e., the computations they have to perform are nearly the same. See Section 4.6 for such a multiparty computation protocol and further details.

4. Alice wanting to do a consumer-opinion poll is an example of Variant 3. Here, Alice collects input values of Bob (and other entities) to evaluate them privately.

Clearly Scenario 1 can be solved with general protocols for secure function evaluation (see e. g., [85, 46]). However, we would like to have a solution that is non-interactive, i. e., Alice and Bob should have only one round of data exchange and should not have to communicate further during the evaluation of the function. Furthermore, it would be optimal to provide a solution in which the workload that Alice has to perform depends only on the efficiency of the underlying encryption scheme, but not on the "size" of f, i. e., the size of a circuit computing f.
Some non-interactive solutions have been described in the literature, see e. g. [77, 16]. However, none of them offers independency of the "size" of f. The best known non-interactive solution to our scenario is described by Sander et al. in [77]. They described a protocol using novel constructions which requires only one round of interaction to evaluate NC^1 circuits and which gives Bob computational privacy for his input y and Alice information-theoretic privacy for her function f. However Alice's workload depends exponentially on the depth d of the circuit C. Therefore, this solution is bounded to circuits of logarithmic depth - known as NC^1 circuit.
Obviously, a natural candidate to solve both requirements (no interaction and independency of the size of f) would be a probabilistic, algebraically homomorphic encryption scheme since it provides a tool for computations on encrypted data. However, an efficient and secure algebraically homomorphic scheme has not been found yet. As already mentioned in Section 2.1 all known schemes have to deal with an exponential blowup during repeated computations. That is why their application scope is restricted to functions that can be implemented by NC^1 circuit. In [5] it was proven that any Boolean circuit of logarithmic depth can be efficiently simulated by a circuit over an arbitrary finite nonsolvable group. Hence the proposed encryptions of non-abelian groups of Grigoriev et al. in [49, 50] (see Section 2.1) can be applied to obtain encryptions of NC^1 circuits. However owing to the exponential blowup of their scheme they cannot do any better than NC^1 circuits. Thus, another approach is necessary to enlarge the class of encryptable functions.

Our solution presented below for Scenario 1 is based on an idea of Ronald Cramer. It has two ingredients. We assume that the function f is given by a branching program in the form of the corresponding adjacency matrix. Then we encrypt that matrix using a probabilistic, additively and scalar homomorphic encryption scheme (see Chapter 4 for an example).

To prove the security of our solution formally we have to guarantee that R is a finite field, although in practice we usually may assume that the ring $\mathbb{Z}/N\mathbb{Z}$ where N is the product of two large primes behaves like a finite field (see Remark 3.2.3). Thus we restrict the considered scenario to the case where the function $f : R^n \to R$ is defined on a finite field R. Furthermore, since all known efficient additively homomorphic encryption schemes are defined on $\mathbb{Z}/k\mathbb{Z}$ for some $k \in \mathbb{N}$ it is wise to restrict R to being a prime field. Thus we may use probabilistic, additively homomorphic cryptosystems as the Goldwasser-Micali scheme [45] on $\mathcal{M} = (\mathbb{Z}/2\mathbb{Z}, +)$, or schemes as the Okamoto-Uchiyama scheme [65] on $\mathcal{M} = (\mathbb{Z}/p\mathbb{Z}, +)$ with p being prime. If additively and scalar homomorphic cryptosystems will be developed on arbitrary finite fields - rather than prime fields - then our solution can immediately be generalised.

Remark 3.2.3. It is also possible to consider functions $f : R^n \to R$ where $R = \mathbb{Z}/N\mathbb{Z}$ and $N = pq$ is the product of two large primes. Given $x \in \mathbb{Z}/N\mathbb{Z}$ the probability that $\gcd(x, N) = 1$ is $\frac{\phi(N)}{N} = \prod_{p|N}(1 - \frac{1}{p}) = (1 - \frac{1}{p})(1 - \frac{1}{q})$. Since this probability is negligibly close to 1, almost all elements are invertible. Thus, in practice we may assume that $\mathbb{Z}/N\mathbb{Z}$ behaves like a finite field. However, in this case the security of our protocol can not be formally proven. Examples of probabilistic, additively homomorphic schemes with $\mathcal{M} = \mathbb{Z}/N\mathbb{Z}$ are Paillier's scheme [67] and the elliptic curve Paillier scheme [42] which we will describe in Section 4.2.

Notation 3.2.4. Let M be a matrix over the message space $\mathcal{M}$ and $E_B : \mathcal{M} \to \mathcal{C}$ be an encryption function under a public key $k_e = B$. The notation $E_B(M)$ means that E_B is applied on each entry of M and that the result is written in a matrix again. When applying the algorithms Add or Mixed-Mult on such ciphertext matrices, we apply them on each entry separately.

For a better understanding we first give a basic version of our solution, and then improve its efficiency afterwards.

Basic Solution:

1. Bob encrypts his input values $y_j, 1 \leq j \leq m$ with his public key B using a probabilistic, additively homomorphic encryption scheme $E : \mathcal{M} \to \mathcal{C}$ where $\mathcal{M} = (R, +)$ and R is a prime field. We denote such encryptions by $E_B(y_j)$ where $1 \leq j \leq m$. Bob sends these encryptions to Alice.

2. To compute an output $f(x, y)$ that can only be read by Bob given his encrypted inputs $E_B(y_j), 1 \leq j \leq m$ and Alice's inputs $x_i, 1 \leq i \leq k$, Alice does the following:

   (a) Let $BP = (G, w, 1 = s, \lambda = t)$ be a branching program of size λ computing Alice's function f. Let H be the corresponding $\lambda \times \lambda$ adjacency matrix. The entries of H are degree-1 polynomials over R in the input variables x_i and y_j, i. e., they equal $\sum_{i=1}^{n} a_{li} z_i + b_l$, $a_{li}, b_l \in R,\ 1 \leq l \leq \lambda^2$ where $z_h \in \{x_i, y_j | 1 \leq i \leq k, 1 \leq j \leq m\},\ 1 \leq h \leq n$ (see Section 3.1). Alice deletes the first column and the last row of H and obtains a matrix H^-. She replaces each entry $\sum_{i=1}^{n} a_{li} z_i + b_l$ of H^- with its encrypted entry $E_B(\sum_{i=1}^{n} a_{li} z_i + b_l)$ to obtain an encrypted matrix $E(H^-)$. Note that this can be done owing to the homomorphic property of the encryption scheme: If $z_h \in \{x_i | 1 \leq i \leq k\}$, i. e., if the entry of the matrix H^- is a polynomial in her input, then Alice is able to compute $E_B(a_{lh} z_h)$ directly using Bob's public key B. Note that Alice knows a_{lh} as part of her function f. If $z_h \in \{y_j | 1 \leq j \leq m\}$, i. e., the entry depends on Bob's input, then given encryptions $E_B(y_j)$ of Bob's inputs an encryption $E_B(a_{lh} z_h)$ can be computed using the algorithm Mixed-Mult of the encryption scheme to obtain

   $$\text{Mixed-Mult}(a_{lh}, E_B(y_j)) = E_B(a_{lh} y_j) = E_B(a_{lh} z_h),$$

   for $1 \leq h \leq n,\ 1 \leq l \leq \lambda^2$. Then given the encryptions $E_B(a_{lh} z_h)$ she encrypts each b_l and uses the algorithm Add to get an encryption $E_B(\sum_{i=1}^{n} a_{li} z_i + b_l)$. Thus she is able to compute an encryption $E(H^-)$ of her (reduced) adjacency matrix (see Section 3.1).

   (b) Alice deletes the first column and the last row of the $\lambda \times \lambda$ identity matrix I to obtain the matrix I^-. Then she computes an encryption $E_B(I^-)$ of I^- by encrypting each entry 0 and 1 using Bob's public key.

   (c) Due to the homomorphic property of the encryption scheme Alice is able to compute $E_B(I^- - H^-) =: E_B(M)$ entry by entry (see Section 3.1) since

$$\text{Add}(E_B(I^-), \text{Mixed-Mult}(-1, E_B(H^-))) = E_B(I^- - H^-) = E_B(M)$$

where the algorithms Add and Mixed-Mult are applied on each entry.

(d) To blind, i. e., to hide the entries $(E_B(M))_{i,j}$ of this matrix, Alice chooses two random $(\lambda - 1) \times (\lambda - 1)$ matrices S, T over R, so that $\det(S)$ and $\det(T)$ are units in R. If f is not a Boolean function then Alice chooses S, T with $\det(S) = (\det(T))^{-1}$. Then she computes $E_B(S \cdot M \cdot T)$ Due to the additively and scalar homomorphic property of the encryption scheme this computation can be done: It is

$$(E_B(SMT))_{i,j} = E_B(\sum_l \sum_k s_{i,k} \cdot m_{k,l} \cdot t_{l,j}) = E_B(\sum_l \sum_k s_{i,k} \cdot t_{l,j} \cdot m_{k,l}).$$

Alice uses the algorithm Mixed-Mult to compute the values

$$\text{Mixed-Mult}(s_{i,k} \cdot t_{l,j}, E_B(m_{k,l})) = E_B(s_{i,k} \cdot t_{l,j} \cdot m_{k,l}).$$

Note that Alice knows the values $(s_{i,k} \cdot t_{l,j}) \in R$. Then she uses the algorithm Add to obtain $E_B(SMT))_{i,j}$. Since R is a field $E_B(S \cdot M \cdot T)$ is an encryption of a random matrix of the same rank as M (see Lemma 3.2.7).

(e) Alice sends the encryption $E_B(S \cdot M \cdot T)$ to Bob.

3. To obtain $f(x, y)$ Bob uses his secret key to decrypt the entries of $E_B(SMT)$ and thus he obtains SMT.
   If f is a Boolean function it is sufficient to compute the rank of this decrypted matrix using an efficient algorithm. If it equals $\lambda - 1$, the output of $f(x, y)$ equals 1, otherwise 0 (see Remark 3.1.3).
   If f was defined over a prime field $\mathbb{Z}/p\mathbb{Z}$ with $p \neq 2$ then Bob applies an efficient algorithm to compute

$$\det SMT = \det M = (-1)^{\lambda+1}(I - H)^{-1}_{1,\lambda} = f(x)$$

to obtain the output of the function (see Section 3.1). Note that it holds that $\det(S) = (\det(T))^{-1}$.

Remark 3.2.5. If $E_B(M)$ was nonsingular then one random nonsingular matrix would have been sufficient in Step 2d to blind the entries $(E_B(M))_{i,j}$, but since Alice does not know the secret key, she is not able to decrypt and to compute the rank.

The following image summarises the idea of our solution:

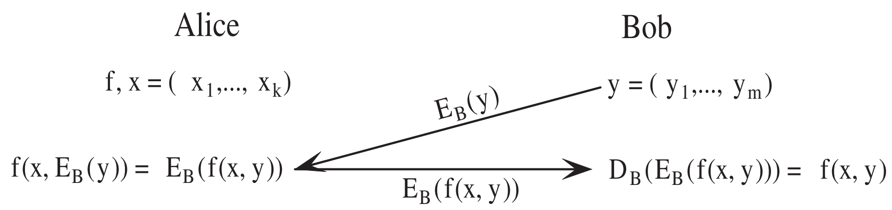


Figure 3.1: Computing with encrypted data

In other words the basic idea of our solution is the following: Alice has a private function f and a private input x. Bob has a private input y that he encrypts using his public key B and sends the encryption to Alice. She evaluates her function f on her private input and Bob's encrypted input. She modifies the result to hide her function and obtains an encrypted "function value" $E_B(f(x, y))$. She sends this value back to Bob. He is now able to decrypt it using his secret key and to compute the output $f(x, y)$.

Remark 3.2.6. A similar solution can be given for counting and counting modulo-p branching programs. These branching programs compute functions $f : \{0, 1\}^n \to \mathbb{N}$ and $f : \{0, 1\}^n \to \mathbb{Z}/_{p\mathbb{Z}}$, respectively (see [53, 54] and [55] for definitions).

To improve efficiency of our solution we slightly modify it in the following way:
We keep the cleartext values as long as possible to reduce the computational complexity since usually additions and multiplications of cleartext values are more efficient than the usage of the algorithms Add and Mixed-Mult on encrypted values.

Improved solution:

1. As before Bob encrypts his inputs $y_j, 1 \le j \le m$ with his public key using a probabilistic, additively homomorphic encryption scheme, and sends these encryptions $E_B(y_j)$ to Alice.

2. To compute an output $f(x, y)$ that can only be read by Bob given the encrypted inputs $E_B(y_j),\ 1 \le j \le m$ and Alice's inputs $x_i,\ 1 \le i \le k$, Alice now does the following:

   (a) First, Alice deletes the first column and the last row of the adjacency matrix H and obtains a matrix H^-. Then she replaces every addend $a_{lh}z_h$ where $z_h \in \{y_j | 1 \le j \le m\}$ of the entry $\sum_{i=1}^{n} a_{li}z_i + b_l$ in H^- with its encrypted entry $E_B(a_{lh}z_h)$ using Bob's public key and the algorithm Mixed-Mult. Alice keeps the addends b_l and the addends $a_{lh}z_h$ where $z_h \in \{x_i | 1 \le i \le k\}$ as they are. Denote by $\tilde{E_B}(H^-)$ the matrix she obtains. Since this matrix is not entirely encrypted we do not denote it by $E_B(H^-)$.

   (b) Alice deletes the first column and the last row of the $\lambda \times \lambda$ identity matrix I to obtain the matrix I^-.

   (c) Then she computes $\tilde{E_B}(I^- - (H^-)) = \tilde{E_B}(M)$ entry by entry. This is done by using the algorithms Add and Mixed-Mult to compute the encrypted values whereas the remaining cleartext values in this matrix can be computed directly. Thus some of the matrix's entries are encrypted, some are partly encrypted and the remaining entries are cleartext values.

   (d) To blind her matrix H Alice chooses two random $(\lambda - 1) \times (\lambda - 1)$ matrices S, T over R, so that $\det(S)$ and $\det(T)$ are units in R with $\det(S) = (\det(T))^{-1}$. She can now compute $E_B(SMT)$: As before we have

   $$(E_B(SMT))_{i,j} = E_B(\sum_l \sum_k s_{i,k} \cdot t_{l,j} \cdot m_{k,l}).$$

   Now some of the values $m_{k,l} \in \tilde{E_B}(M)$ are encrypted and others are partly encrypted or cleartext values. To compute $(E_B(SMT))_{i,j}$ she computes the products and sums of the cleartext values and uses the algorithms Mixed-Mult and Add to compute the other products and sums. After that she encrypts the remaining cleartext values of this matrix and uses the algorithm Add to obtain the same matrix $E_B(SMT)$ as in the above basic solution.

(e) She sends $E_B(SMT)$ to Bob.

3. Bob does the same as above to obtain $f(x, y)$.

Efficiency:
Considering the communication complexity our protocol is optimal since it is non-interactive, i. e., it needs just one round of communication. Concerning the computational complexity it depends mainly on the size of the underlying branching program as well as on the efficiency of the homomorphic cryptosystem. Further computations like the selection of random matrices in Step 2d or the computation of the determinant or rank in Step 3 can be done efficiently. Obviously, the efficiency of this protocol is restricted to functions that are computable by polynomial-size branching programs. The class of such functions that are computable by branching programs of polynomial size includes the class of NC^1 functions. Since known results in the literature are restricted to NC^1 functions, our result is an improvement (see e. g. [77]).

Correctness:
The correctness of our protocol follows immediately from the results presented in Section 3.1, from the properties of the algorithms Add and Mixed-Mult, from the multiplicativity of the determinant, i. e., $\det SMT = \det S \cdot \det M \cdot \det T$ as well as from the fact that multiplication by random nonsingular matrices does not change the rank.

Security:
One possible "attack" that Alice is able to perform is *denial of service*, by not sending any value back to Bob. However, in scenarios where Alice wants to get paid for sending Bob the encrypted output, this attack is unlikely.
Another attack would be that Alice sends a wrong result to Bob. This is a general problem of such protocols although in many scenarios Bob is able to verify the

correctness of the output. To avoid such an attack Bob could ask Alice for a zero-knowledge proof to verify the correctness of her computation (see Remark 3.3.4).
What we consider here is security concerning the privacy of Bob's inputs and Alice's function (and her inputs as part of her function). It follows immediately that the privacy of Bob's inputs is equivalent to the security of the underlying encryption scheme since Alice only obtains Bob's encrypted inputs. If the homomorphic cryptosystem is semantically secure such as, e. g. Paillier's scheme, then Alice is not able to gain any information about Bob's inputs. Furthermore, since Alice is not able to decrypt the entries of the matrices, she is not able to compute the rank or the determinant of SMT and thus of M, i. e., she is not able to compute the output $f(x, y)$.
Bob receives an encryption of a random matrix SMT of a fixed rank and of a fixed determinant. Hence he is not able to derive any information about the adjacency matrix H and thus about Alice's function f - except its size. (Note that we can also provide a protocol in which he only gets to know un upper bound of this size, see Section 3.3). Furthermore, he is not able to decide which entries belong to Alice's inputs. This argument why Alice's function remains private will be further formalized in the following.

We establish the privacy of Alice's function through the following lemma. It shows that the matrix $E_B(S \cdot M \cdot T)$ that Alice obtains after blinding her matrix is an encryption of a random matrix that has the same rank and the same determinant as M. The entries of this matrix $E_B(SMT)$ are the only values that Bob receives from Alice. Hence, if this matrix is an encryption of a random matrix, Bob is not able to gain any extra information about Alice's function - except its size.

Lemma 3.2.7. Let M be a $n \times n$ matrix over $\mathbb{F}_q$ and let S and T be uniform-random nonsingular $n \times n$ matrices with $\det(T) \cdot \det(S) = 1$ over $\mathbb{F}_q$. Then for fixed M each possible matrix SMT occurs with the same probability.

Proof. Consider the group $G = \{(S, T) \mid \det(T) \cdot \det(S) = 1\} \subseteq GL(n, q) \times GL(n, q)$. This group operates on the set $\mathbb{M} := \{M \mid M \text{ is a } n \times n \text{ matrix over } \mathbb{F}_q\}$ as $M \mapsto S \cdot M \cdot T$. Note that this operation does not change the determinant and rank of $M \in \mathbb{M}$. Consider the supgroup $G_{\mathbb{M}}$ of G with $G_{\mathbb{M}} = \{(S, T) \in G \mid SMT = M\}$

and for $M \in \mathbb{M}$ the subset $G.M := \{SMT \mid (S,T) \in G\}$ of $\mathbb{M}$. Then the mapping

$$G/G_{\mathbb{M}} \to G.M,\ (S,T) \cdot G_{\mathbb{M}} \mapsto SMT$$

is a bijection. Due to the equipotency of the orbits $G.M$ and the uniform distribution of S and T the claim follows. □

In our protocol it is essential that the homomorphic scheme is probabilistic which guarantees privacy for Bob's input since Alice is not able to compare different encryptions. This condition is already included in our security analysis above since we require a semantically secure encryption scheme. Hence, our protocol is as secure as the probabilistic, homomorphic cryptosystem used.

Remark 3.2.8. If f is a polynomial of degree d that Alice wants to hide, she should not output more than d values to Bob (or someone else). Otherwise Bob could use the function values to obtain Alice's polynomial by Lagrange's interpolation.

3.3 Computing with Encrypted Functions

We now have a closer look at the closely related problem called *computing with encrypted functions*. The scenario is the following:

Scenario 2:
Alice has a function $f : R^n \to R$ and Bob has a secret input $x = (x_1, \ldots, x_n)$. Bob is willing to compute $f(x)$ locally on his input x in such a way that only Alice learns the output of the function. Bob and Alice should learn nothing about the other ones input besides that what the output reveals. In particular, Alice's function should remain secret.

Note that depending on the function f the output $f(x)$ could reveal the entire input x.

It is also possible to consider the following two variants of this scenario:

Variants:

1. Only Bob should learn the output $f(x)$.

2. Alice and Bob should learn the output $f(x)$.

Example 3.3.1.

1. As an example of Scenario 2 we could think about a mobile agent f that collects data x. In this case we would think about Bob's input x as publicly known. For instance, Alice wants to book a flight and searches for the cheapest flight at her desired time. For this purpose she sends a mobile agent to the servers of different airlines who collects the appropriate data to find out the cheapest flight. After that it returns to Alice. To protect such an agent against malicious hosts it is encrypted. See [76] for further details.
   If the mobile agent also tells the airline servers the cheapest flight, Variant 2 is implemented.

2. As an example of Variant 1 of Scenario 2 we could think about an efficient algorithm for factoring that Alice wants to sell to Bob without providing him with the algorithm. In contrast to Example 3.2.2 this time it should be Bob who performs most of the computations.

A similar scenario was mentioned by several authors, but with slightly different notions. For instance, Abadi and Feigenbaum [1] considered a circuit that implements the function f instead of considering f itself. Such a scenario is called *secure circuit evaluation.* Furthermore, Abadi and Feigenbaum pointed to the following relation between computing with encrypted data (see Scenario 1) and computing with encrypted functions (see Scenario 2):

"*It is also clear from this description that the distinction between "data" and "circuits" is unnecessary. If [Alice] has the ability to hide a circuit, then [she] can also hide some private data, simply by "hardwiring" it into the circuit. Conversely, in protocols in which [Alice] has the ability to hide data, [she] can also hide a circuit through a detour: [Alice] can run the protocol, take the circuit for f to be a universal circuit, and use an encoding of the circuit [she] wants to hide as input.*"

Since we consider branching programs as a computational model for functions instead of circuits, this result can not directly be applied to our case. Furthermore, a "universal branching program" has not been defined yet.

We solve Scenario 2 for functions $f : R^n \to R$, where R is a prime field in the following way. Note that as before in practice our solution can be generalised to rings $R = \mathbb{Z}/N\mathbb{Z}$ where $N = pq$ but where there is no formal proof of security known.

Solution:

1. Alice encrypts the adjacency matrix H belonging to the branching program that computes the function f. To do so, she uses a probabilistic, additively and scalar homomorphic encryption scheme on R and encrypts the entries of H using her public key A. She obtains a "matrix"

$$E_A(H) := \begin{pmatrix} 0 & E(a_1^{1,2}),\ldots,E(a_n^{1,2}),E(b_{1,2}) & \ldots & E(a_1^{1,\lambda}),\ldots,E(a_n^{1,\lambda}),E(b_{1,\lambda}) \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & E(a_1^{\lambda-1,\lambda}),\ldots,E(a_n^{\lambda-1,\lambda}),E(b_{\lambda-1,\lambda}) \\ 0 & 0 & \ldots & 0 \end{pmatrix}$$

where $w(i,j) = \sum_{k=1}^{n} a_k^{i,j} x_k + b_{i,j}$ and λ is the size of the branching program. If (i,j) is not an edge in G she sets $a_k^{i,j} = b_{i,j} = 0$ for $1 \leq k \leq n$ and encrypts these values.
Alice sends $E_A(H)$ to Bob. Hence, Bob gets to know the size of the branching program. If we only want him to learn an upper bound of the size then we can use the matrices U and N defined below in Remark 3.3.2 instead of the matrices H and M.

2. Bob uses the algorithms Add and Mixed-Mult to compute the matrix

$$E_A(H(x)) = E_A(H)(x) = \begin{pmatrix} 0 & E(\sum_{i=1}^{n} a_i^{1,2} x_i + b_{1,2}) & \ldots & E(\sum_{i=1}^{n} a_i^{1,\lambda} x_i + b_{1,\lambda}) \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & E(\sum_{i=1}^{n} a_i^{\lambda-1,\lambda} x_i + b_{\lambda-1,\lambda}) \\ 0 & 0 & \ldots & 0 \end{pmatrix}$$

where $x = (x_1, \ldots, x_n)$ is his private input. He then computes an encryption $E_A(I)$ of the $(\lambda-1) \times (\lambda-1)$ identity matrix I using Alice's public key. Let $E_A(H(x)^-)$ denote the matrix obtained by deleting the first column and the last row of $E_A(H(x))$. Bob is now able to compute

$$E_A(M) = E_A(I - H(x)^-) = \mathrm{Add}(E_A(I), \text{Mixed-Mult}(-1, E_A(H(x)^-))).$$

To blind the entries of $E_A(M)$ and thus to hide his secret input x he chooses two random nonsingular $(\lambda-1)\times(\lambda-1)$ matrices S, T over R with $\det(S) = (\det(T))^{-1}$ and computes $E_A(S \cdot M \cdot T)$. As already described in Section 3.2 this computation can be done using the algorithms Add and Mixed-Mult of the underlying encryption scheme. Bob sends $E_A(S \cdot M \cdot T)$ to Alice.

3. To obtain $f(x)$ Alice has to compute $(-1)^{\lambda+1}$ multiplied with the determinant or if f is Boolean the rank of the decryption of the matrix $E_A(S \cdot M \cdot T)$: She uses her secret key to decrypt the entries and applies an efficient algorithm that computes the determinant and rank, respectively.

The image visualizes the idea of our protocol:

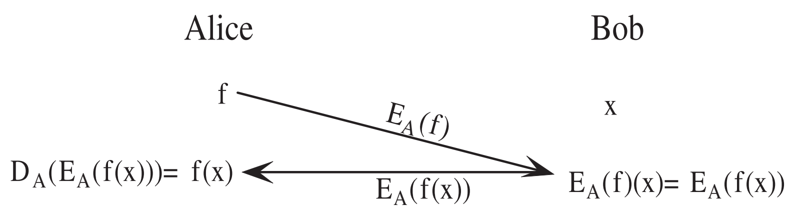


Figure 3.2: Computing with encrypted functions

Hence, the basic idea of our solution is as follows: Alice has a function f that she encrypts with her public key A and sends the encryption $E_A(f)$ to Bob. Bob evaluates this encrypted function on his secret input x in such a way that he obtains an encryption $E_A(f(x))$. Furthermore, he blinds his input and sends the result back to Alice. Alice is now able to decrypt it using her secret key and to compute the output of her function $f(x)$.

Remark 3.3.2. Define a matrix

$$U = \begin{pmatrix} O & O \\ O & H \end{pmatrix},$$

where H is the adjacency matrix belonging to a branching program that computes the function f, and O is a $r \times r$ zero-matrix. If H is a $\lambda \times \lambda$ matrix then U is a $(r + \lambda) \times (r + \lambda)$ matrix. Since H is nilpotent U is nilpotent, too. It holds that

$$U^i = \begin{pmatrix} O & O \\ O & H^i \end{pmatrix}$$

and we define $U^* := \sum_{k=0}^{\infty} U^k$. As before (see Section 3.1) we have $U^* = (I - U)^{-1}$ and $\det(I-U) = 1$. Since the entry $(1, \lambda)$ of $H^* = \sum H^k$ equals the entry $(r+1, r+\lambda)$ of U^*, we have

$$U^*_{r+1,r+\lambda} = W(1, \lambda)(x) = \frac{\det N}{\det(I - U)} = \det N = f(x),$$

where N denotes the submatrix of $I - U$ obtained by deleting the $(r + 1)$-th column and the $(r + \lambda)$-th row. Therefore, the function value $f(x)$ can be determined by computing the determinant of a larger sized matrix N instead of computing the determinant of M (multiplied by $(-1)^{r+\lambda+1}$). Hence, instead of using H Alice could also use the matrix U in our protocol. When encrypting U she would encrypt the whole upper triangular matrix, i. e., including the zeros in the upper triangular matrix of U. Then Bob has to compute an encryption of N instead of M and has to change the sizes of the matrices S and T adequately.
The advantage of using U and N instead of H and M is that Bob just learns an upper bound for the branching program size and not the exact size. Of course, this is at the cost of additional computing power since the matrices are larger than before.

Remark 3.3.3. In [26] Cramer et al. proposed a way to garble branching programs which are defined over arbitrary rings. Their technique can also be applied to our protocol. Since a garbled branching program has a full matrix instead of an upper triangular matrix as adjacency matrix this would lead to a less efficient protocol with a slightly changed semantic. Therefore this technique should not be used in our protocol.

Security, Correctness, Efficiency:
The correctness, efficiency and security analysis of this protocol is exactly the same as of the one in Section 3.2. Hence, as before our protocol is provably secure, non-interactive and efficient for polynomial-size branching programs. This is an improvement to known result in the literature.

Remark 3.3.4.

1. It is sufficient if the encryption algorithm E is a symmetric algorithm and remains secret. For the computations Bob only needs the algorithms Add and Mixed-Mult. Note that if E is symmetric then Alice has to send an encryption $E(I)$ of the identity matrix I, too.

2. Alice could ask Bob for proving, e. g. in zero-knowledge fashion (see Section 4.5.1), that he knows his input x, that x is a valid input, and that the function $E(f)$ evaluated on x equals $E(f(x))$. See [22] for such protocols.

3.4 Key Swapping

In this section we introduce a new property that we call *key swapping*. We shortly demonstrate why this property of cryptosystems is useful, especially in the context of this chapter. Furthermore, we show that there already exist multiplicatively as well as additively homomorphic, probabilistic cryptosystems having this property.

Definition 3.4.1. Given a cryptosystem (K, E, D, A) on the message space $\mathcal{M}$ where for the ciphertext space it holds that $\mathcal{C} \subseteq \mathcal{M}$. Let $E_{k_e}(m)$ denote an encryption of a message $m \in \mathcal{M}$ encrypted with a public key k_e. Let a be the public key of a party A and b be the public key of another party B. The cryptosystem is called *key swapping* if there exists an efficient algorithm Key-Swap that given an encryption $E_a(E_b(m))$ computes an encryption $E_b(E_a(m))$.

A probabilistic, additively and scalar homomorphic encryption scheme that satisfies key swapping could be used to solve the following Variant 1 of Scenario 2:

Alice has a function $f : R^n \to R$ and Bob has an input $x \in R$ for f that he wants to evaluate. Bob and Alice should learn nothing about the other ones inputs, i. e., f and x should remain secret and only Bob should learn the output $f(x)$. Furthermore, Bob should do most part of the computation on his own.

Given a probabilistic, additively and scalar homomorphic cryptosystem that satisfies key swapping, a solution similar to the ones described in Section 3.2 and Section 3.3 can easily be given. The idea is displayed in the following figure:

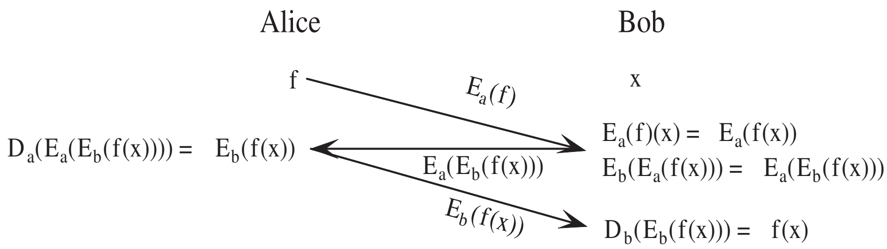


Figure 3.3: Key swapping

Alice encrypts her function f (i. e., the corresponding branching program) by her public key a. She sends the encryption $E_a(f)$ to Bob, who evaluates it on his input x to obtain $E_a(f(x))$. He uses his public key b to encrypt the result $E_a(f(x))$ and obtains $E_b(E_a(f(x))) = E_a(E_b(f(x)))$ which he sends back to Alice. Alice now uses her secret key to decrypt it and obtains $E_b(f(x))$ which she sends back to Bob who is able to decrypt it and to obtain the desired function value $f(x)$.
Obviously Variant 3 of Scenario 1 can be solved analogously.

Remark 3.4.2. Another possibility to solve these scenarios is the following: Given a function $f : R^n \to R$ where R is a finite ring. To hide the function value $f(x)$ for $x = (x_1, \ldots, x_n) \in R^n$ we may compute $f(x) + r$ where $r \in R$ is chosen at random. Then $f(x) + r$ is a random value in R, too.
For instance, in Variant 1 of Scenario 2 Alice could encrypt her function f with a homomorphic cryptosystem using her public key a and send $E_a(f)$ to Bob. If $E_a(f)(x) = E_a(f(x))$ holds, Bob can evaluate the function on his private input x. Furthermore, he can choose a random value r and compute $\mathrm{Add}(E_a(f(x)), E_a(r)) =$

$E_a(f(x) + r)$. If he sends this to Alice she is able to decrypt it to obtain $f(x) + r$ which is a random value. If she sends it back to Bob he is able to subtract r to obtain the function value $f(x)$.
In an analogous manner Variant 3 of Scenario 1 can be solved. There Bob encrypts his input x with a homomorphic cryptosystem using his public key b and sends the result $E_b(x)$ to Alice. If $f(E_b(x)) = E_b(f(x))$ holds, Alice can evaluate $f(x) + r$, where she chooses r at random, on $E_b(x)$ to obtain $E_b(f(x) + r)$. If she sends this value back to Bob he is able to decrypt it in order to obtain $f(x) + r$ which is again a random value. If he sends $f(x) + r$ back to Alice she can subtract her secret, random value r to obtain the output $f(x)$.

3.4.1 Examples

In this section we present two examples of homomorphic schemes that satisfy key swapping.

The ElGamal Scheme

The ElGamal scheme [37] was first published in 1984. It is based on the difficulty of the computation of discrete logarithms. It is a popular example of a probabilistic, multiplicatively homomorphic public-key scheme. Furthermore, it supports key swapping as we will show below.

The ElGamal cryptosystem works as follows:

Key Generation:

- Generate a large random prime p and a generator g of the multiplicative group $(\mathbb{Z}/p\mathbb{Z})^*$.
- Select a random integer $x, 1 \leq x \leq p - 2$ and compute $y = g^x \mod p$.

The public key of A is $a = (p, g, y)$ and the secret key is x.

Encryption:
To encrypt a message $m \in \mathcal{M} = (\mathbb{Z}/_{p\mathbb{Z}})^*$ select a random integer $1 \leq r \leq p-2$ and compute

$$c = (\gamma, \delta) = (g^r \bmod p,\ m \cdot y^r \bmod p) =: E_a(m, r).$$

Decryption:
To decrypt $E_a(m, r) = (\gamma, \delta) \in\ \mathcal{C}$ where $\mathcal{C} = (\mathbb{Z}/_{p\mathbb{Z}})^* \times (\mathbb{Z}/_{p\mathbb{Z}})^*$ compute

$$\gamma^{-x} = \gamma^{p-1-x} = g^{-xr} \bmod p$$

then

$$m = (\gamma^{-x}) \cdot \delta \bmod p.$$

Homomorphic Property:
This scheme is multiplicatively homomorphic: If

$$E_a(m_1, r_1) = (\gamma_1, \delta_1) = (g^{r_1} \bmod p, m_1 \cdot y^{r_1} \bmod p)$$

and

$$E_a(m_2, r_2) = (\gamma_2, \delta_2) = (g^{r_2} \bmod p, m_2 \cdot y^{r_2} \bmod p)$$

then

$$E_a(m_1, r_1) \cdot E_a(m_2, r_2) := (g^{r_1+r_2} \bmod p, m_1 \cdot m_2 \cdot y^{(r_1+r_2)} \bmod p) = E_a(m_1 \cdot m_2, r_1 + r_2).$$

Hence the algorithm Mult can simply be implemented by multiplying the corresponding plaintexts and using a blinding algorithm (see Definition 2.0.7).

Key Swapping:
Given the public keys $a = (p, g, y)$ of a party A and $b = (p, g', y')$ of a party B where $y = g^x \bmod p$ and $y' = g'^{x'} \bmod p$ with corresponding secret keys x and x', respectively. Note that the prime p should be the same in both keys, i. e., we are working in the same group, but the generator g may be chosen differently. We first have to define the encryption under the public key b of a given ciphertext $c = (g^r \bmod p, m \cdot y^r \bmod p) = E_a(m, r)$, i. e., we have to extend the message space

from $(\mathbb{Z}/p\mathbb{Z})^*$ to $(\mathbb{Z}/p\mathbb{Z})^* \times (\mathbb{Z}/p\mathbb{Z})^*$:

$$\begin{aligned} E_b(c, r') = E_b(E_a(m, r), r') &= E_b((g^r \bmod p, m \cdot y^r \bmod p), r') \\ &:= (g'^{r'} \bmod p, g^r \bmod p, m \cdot y^r y'^{r'} \bmod p) \\ &= (g'^{r'} \bmod p, g^r \bmod p, m \cdot g^{xr} g'^{x'r'} \bmod p). \end{aligned}$$

Now we can define the algorithm Key-Swap:

$$\begin{aligned} \text{Key-Swap}(E_a(E_b(m, r')), r) &= \text{Key-Swap}(g^r \bmod p, g'^{r'} \bmod p, m \cdot g^{xr} g'^{x'r'} \bmod p) \\ &= (g'^{r'} \bmod p, g^r \bmod p, m \cdot g^{xr} g'^{x'r'} \bmod p) \\ &= E_b(E_a(m, r), r'). \end{aligned}$$

So, given $E_a(E_b(m, r'), r)$ the algorithm Key-Swap just swaps the first two coordinates of this triple to obtain $E_b(E_a(m, r), r')$. This can obviously be done extremly efficiently.
Given an encryption $E_b(E_a(m, r), r')$ the corresponding secret key to the public key b can be used to obtain $E_a(m, r)$. This is straightforward. An analysis of the security of the ElGamal scheme and further details can be found in [37].

The Damgård-Jurik Scheme

Now we give another example of an encryption scheme that is key swapping. This time it is a probabilistic, additively and scalar homomorphic cryptosystem and hence can be used to solve the scenarios presented in the last sections. It was recently published by I. Damgård and M. Jurik [31]. The scheme works as follows:

Key Generation:

- Choose an RSA modulus $N = pq = (2p' + 1)(2q' + 1)$ with primes p, p', q, q'.
- Select an element $g \in Q_N$ where Q_N denotes the group of all squares in $(\mathbb{Z}/N\mathbb{Z})^*$. Choose $\alpha \in \mathbb{Z}/\tau\mathbb{Z}$, where $\tau = p'q' = |Q_N|$.
- Compute $h = g^\alpha \bmod N$.

The public key is $a := (N, g, h)$ and the secret key is α.

Encryption:
To encrypt a message m, choose an integer $s > 0$ so that $m \in \mathbb{Z}/_{N^s\mathbb{Z}}$, and choose a random $r \in \mathbb{Z}/_{n\mathbb{Z}}$ where $n = 4^{\log_2 N}$. The ciphertext is

$$E_a(m, r) = (g^r \bmod N, (h^r \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}) =: (G, H).$$

Let L_s denote a function with

$$L_s((N+1)^m \bmod N^{s+1}) = m \bmod N^s.$$

An algorithm that computes this function, i. e., that calculates the discrete logarithm with respect to the element $(N+1)$ is described in [30].

Decryption:
Given a ciphertext $c = (G, H) = E_a(m, r)$, s can be deduced from the length of c or it is attached to the encryption. Then the message m can be found as

$$\begin{aligned} m &= L_s(H(G^\alpha \bmod N)^{-N^s}) \\ &= L_s((g^{\alpha r} \bmod N)^{N^s}(N+1)^m(g^{r\alpha} \bmod N)^{-N^s}) \\ &= L_s((N+1)^m \bmod N^{s+1}). \end{aligned}$$

Homomorphic Property:
This scheme is additively homomorphic since given $E_a(m_1, r_1)$ and $E_a(m_2, r_2)$ we can compute

$$\begin{aligned} E_a(m_1 + m_2, r_1 + r_2) &= (g^{r_1+r_2} \bmod N, (h^{r_1+r_2} \bmod N)^{N^s}(N+1)^{m_1+m_2} \bmod N^{s+1}) \\ &= E_a(m_1, r_1) \cdot E_a(m_2, r_2). \end{aligned}$$

Hence the algorithm Add can efficiently be implemented by multiplying the input ciphertexts and applying a blinding algorithm.

Key Swapping:
Given the public keys $a := (N, g, h)$ where $h = g^\alpha$ and $b := (N, g', h')$ where $h' = g'^\beta$ with corresponding private keys α and β, respectively. Note that $N = pq$ must

be the same for both public keys. Then the Damgård-Jurik scheme satisfies key swapping:
We first have to define an encryption of a ciphertext, i. e. we have to enlarge the message space:

$$\begin{aligned} E_a(E_b(m,r'),r) &= E_a(g'^{r'} \bmod N, (h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}) \\ &:= (g^r \bmod N, g'^{r'} \bmod N, (h^r h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}). \end{aligned}$$

Then

$$E_b(E_a(m,r),r') = (g'^{r'} \bmod N, g^r \bmod N, (h^r h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}).$$

Hence, given $E_a(E_b(m,r'),r)$ the algorithm Key-Swap can swap the first two coordinates of this triple to compute $E_b(E_a(m,r),r')$. To obtain $E_a(m,r)$ given an encryption

$$\begin{aligned} E_b(E_a(m,r),r') &= (g'^{r'} \bmod N, g^r \bmod N, (h^r h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}) \\ &=: (F,G,I) \end{aligned}$$

we compute

$$\begin{aligned} I(G^\alpha \bmod N)^{-N^s} &= (h^r h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}(g^{\alpha r} \bmod N)^{-N^s} \\ &= (g^{\alpha r} h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1}(g^{\alpha r} \bmod N)^{-N^s} \\ &= (h'^{r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1} \\ &= (g'^{\beta r'} \bmod N)^{N^s}(N+1)^m \bmod N^{s+1} \\ &=: H. \end{aligned}$$

Hence $(F,H) = E_a(m,r)$ is the desired decryption.

Further properties of this scheme and a proof for its semantic security under two reasonable assumptions are given in [31].

Chapter 4

A Threshold Version of the Elliptic Curve Paillier Scheme and its Applications

The Paillier scheme [67] is an example of a very efficient, probabilistic, additively and scalar homomorphic encryption scheme based on arithmetics in the ring of integers modulo N^2 where N is the product of two large primes. It was published in 1999 and analysed and extended by several authors (such as [31, 19, 30, 18, 20]). One of these extensions is the elliptic curve Paillier scheme, ECPS for short, which was recently published by S. Galbraith [42]. The ECPS is a generalisation of Paillier's encryption scheme from the integers modulo a square to elliptic curves over rings. Paillier himself tried to generalise his scheme to the elliptic curve setting by using anomalous elliptic curves over rings [68], but Galbraith found security flaws in this generalisation [42] whereas the ECPS can be proven semantically secure relative to a new defined problem. In the same way as I. Damgård and M. Jurik managed to generalise the original Paillier scheme to higher modules to enable a wider application scope Galbraith developed a generalisation of the elliptic curve Paillier scheme (see [42]).

The Paillier scheme, the new elliptic curve version by Galbraith as well as his further generalisation are examples for probabilistic, additively homomorphic cryptosystems, which are also scalar homomorphic, i. e., they have the properties that

are necessary for the solutions to the scenarios presented in the last chapter.
The performance of the ECPS and of its generalisation are by far slower than the original Paillier scheme together with the generalisation of Damgård and Jurik since they operate on elliptic curves modulo large numbers. Hence, the elliptic curve version is mainly of theoretical interest. One interesting point is that the elliptic curve version is based on a slightly different assumption than Paillier's original version. This assumption may also hold even if the original Paillier assumption were broken.

In this chapter we begin with giving a short introduction to elliptic curves over rings to recall the basic facts that are needed for the following schemes. Then, after summarising the ECPS and Galbraith's generalisation we develop a further generalisation of the ECPS: We will construct a threshold decryption version in an analogous manner as Damgård and Jurik did based on Paillier's scheme [30]. Thus our new threshold ECPS is a threshold version of the ECPS on one side and an elliptic curve version of the threshold Paillier scheme by Damgård and Jurik on the other side.

$$\begin{array}{ccc} \text{Paillier} & \rightarrow & \text{ECPS} \\ \downarrow & & \downarrow \\ \text{Threshold Paillier} & \rightarrow & \textbf{Threshold ECPS} \end{array}$$

Our threshold version allows any subset of k out of ℓ players to decrypt a ciphertext correctly and efficiently while it disallows any subset of less than k players to obtain any information about the message. Based on our threshold version we are able to describe various applications in a similar way as it was done based on the threshold version of Paillier's scheme (see [31, 24, 32, 33]).
First we explain how a length-flexible variant of the generalised ECPS can be constructed, i. e., a variant that allows to determine the size of the ciphertext during encryption. This is a useful tool for different applications such as mix-nets. Furthermore, we show how the threshold ECPS can be used to devise general multiparty computation protocols, i. e., using threshold decryption we present a multiparty computation protocol which allows to compute an encryption of the product of two messages given the corresponding ciphertexts but not the messages themselves. As

further application we describe how a protocol for electronic voting can be obtained, in which each voter simply posts his ballot on a bulletin board in such a way that the final tally can be computed as sum of all votes while its correctness is verifiable to any observer and while the privacy of the individual votes is guaranteed. As a last application we present a way to build a special commitment scheme based on the elliptic curve Paillier scheme that has some interesting properties, e. g. it can be instantiated in either perfectly hiding or perfectly binding. For most of these applications some further sub-protocols are needed which are described in Section 4.5. These protocols are the first elliptic curve variants of the protocols developed using Paillier's idea.

4.1 Introduction to Elliptic Curves over Rings $\mathbb{Z}/_{N^s\mathbb{Z}}$

We shortly introduce elliptic curves over rings to provide the basic facts that are needed to understand Galbraith's elliptic curve Paillier scheme as well as his generalisation and our further generalisation. This introduction is mainly based on [42]. For further details and a general introduction to elliptic curves we refer to [80].

Let R be a commutative ring with 1 and R^* the set of invertible elements in R. An *elliptic curve*

$$y^2z = x^3 + axz^2 + bz^3$$

over R is defined by a pair $a, b \in R$ such that $6(4a^3 + 27b^2) \in R^*$. As usual the set of R-valued points is denoted by $E(R)$ and is defined to be the set of equivalence classes of points $(x : y : z)$ with $x, y, z \in R$, $y^2z = x^3 + axz^2 + bz^3$ and such that the ideal generated by x, y, z is R, and for which the following equivalence relation holds:

$$(x : y : z) \sim (x' : y' : z') \;\Leftrightarrow\; \exists\, \lambda \in R^* : \lambda x = x', \lambda y = y', \lambda z = z'.$$

For the description of the elliptic curve Paillier scheme and its generalisations we need to consider rings $R = \mathbb{Z}/_{N^s\mathbb{Z}}$ with N being the product of two primes p and q, and $s \in \mathbb{N}$. For such rings the usual chord and tangent operation provides a group law on $E(\mathbb{Z}/_{N^s\mathbb{Z}})$ with $\mathcal{O} = (0 : 1 : 0)$ as identity element. By the Chinese

Remainder Theorem it follows that $E(\mathbb{Z}/_{N^s}\mathbb{Z}) \simeq E(\mathbb{Z}/_{p^s}\mathbb{Z}) \times E(\mathbb{Z}/_{q^s}\mathbb{Z})$. To add two points $(x_1 : y_1 : z_1)$ and $(x_2 : y_2 : z_2)$ on an elliptic curve over $\mathbb{Z}/_{N^s}\mathbb{Z}$ we can use the usual formulae for $z_1, z_2 \in (\mathbb{Z}/_{N^s}\mathbb{Z})^*$. To avoid problems if z_1 or z_2 are not invertible - as it will often occur in our setting - we make the curves affine using the $(x, z)-$plane by requiring that $y = 1$. Note that from z not being a unit in R it follows from the curve equation that x is not a unit, too. Thus in this case y has to be a unit and we can set $y = 1$. Hence, we consider curves of the form $z = x^3 + axz^2 + bz^3$. A point $(x : y : z)$ now becomes $(x/y, z/y)$ and the identity element is represented by $(0, 0)$. The inverse of a point (x, z) is obtained as $(-x, -z)$. As long as the division operations are defined in the ring, the sum (x_3, z_3) of two points (x_1, z_1) and (x_2, z_2) for $x_1 \neq x_2$ can be computed by the following group formulae:

$$\begin{aligned} x_3 &= x_1 + x_2 + \frac{(z_1 - \lambda x_1)(2a\lambda + 3b\lambda^2)}{1 + a\lambda^2 + b\lambda^3}, \\ z_3 &= \lambda(x_3 + x_1) - z_1, \text{ where} \\ \lambda &= \frac{z_1 - z_2}{x_1 - x_2}. \end{aligned}$$

If $x_1 = x_2$ such that we double a point (x_1, z_1) we obtain the point (x_3, z_3) as

$$\begin{aligned} x_3 &= 2x_1 + \frac{(z_1 - \lambda z_1)(2a\lambda + 3b\lambda^2)}{1 + a\lambda^2 + b\lambda^3}, \\ z_3 &= \lambda(x_3 + x_1) - z_1, \text{ where} \\ \lambda &= \frac{3{x_1}^2 + a{z_1}^2}{1 - 3b{z_1}^2 - 2ax_1z_1}. \end{aligned}$$

If the divisions are not defined we need to use other formulae. To obtain such formulae we have to recall some facts about the p-adic theory of elliptic curves, see [80] for further details. Let

$$E_1(\mathbb{Z}/_{N^s}\mathbb{Z}) := \{P \in E(\mathbb{Z}/_{N^s}\mathbb{Z}) : P \text{ reduces to } (0 : 1 : 0) \text{ in } E(\mathbb{Z}/_{N}\mathbb{Z})\}.$$

Thus, for an element $(x : y : z) \in E_1(\mathbb{Z}/_{N^s}\mathbb{Z})$ it holds that $N \mid x, N \mid z$ and $N \nmid y$.[1] Consider the elliptic curve $z = x^3 + axz^2 + bz^3$ over $\mathbb{Z}/_{N^s}\mathbb{Z}$. Given any x with $N \mid x$

[1] It even holds that $N^3 \mid z$.

we find a unique solution where $N \mid z$. It has the form:

$$w(x) = x^3 + ax^7 + bx^9 + 2a^2x^{11} + 5abx^{13} + (5a^3 + 3b^2)x^{15} + 21a^2bx^{17} + \cdots$$

(for example solving the equation by iteration). Since $N \mid x$ the given sum $w(x)$ is finite in $\mathbb{Z}/_{N^s\mathbb{Z}}$ and $N^3 \mid w(x)$.
Let $\psi : N(\mathbb{Z}/_{N^s\mathbb{Z}}) \to E_1(\mathbb{Z}/_{N^s\mathbb{Z}})$ with $x \mapsto (x : 1 : w(x))$. The image of the subgroup $N^j(\mathbb{Z}/_{N^s\mathbb{Z}})$ under ψ is the subgroup

$$E_j(\mathbb{Z}/_{N^s\mathbb{Z}}) = \{(x : 1 : z) \in E_1(\mathbb{Z}/_{N^s\mathbb{Z}}) : N^j|x \text{ and } N^{3j}|z\}$$

of $E_1(\mathbb{Z}/_{N^s\mathbb{Z}})$. Obviously, $E_s(\mathbb{Z}/_{N^s\mathbb{Z}}) = (0 : 1 : 0)$. The map ψ is bijective and has certain homomorphic properties. Although it is not a group homomorphism itself for all s, it induces a group homomorphism from $N^j(\mathbb{Z}/_{N^s\mathbb{Z}})/N^{j+1}(\mathbb{Z}/_{N^s\mathbb{Z}})$ to the group $E_j(\mathbb{Z}/_{N^s\mathbb{Z}})/E_{j+1}(\mathbb{Z}/_{N^s\mathbb{Z}})$. In particular, we get

$$|E_j(\mathbb{Z}/_{N^s\mathbb{Z}})| = N \cdot |E_{j+1}(\mathbb{Z}/_{N^s\mathbb{Z}})|$$

for all $1 \leq j \leq s-1$. Hence $|E_1(\mathbb{Z}/_{N^s\mathbb{Z}})| = N^{s-1}$. Note that for $s = 2$ we obtain an isomorphism between $\mathbb{Z}/_{N\mathbb{Z}}$ and $E_1(\mathbb{Z}/_{N^2\mathbb{Z}})$.
Now we can give an explicit formula for the sum of the points $(x_1 : 1 : w(x_1))$ and $(x_2 : 1 : w(x_2))$. The sum is the point $(x_3 : 1 : w(x_3))$ where

$$\begin{aligned} x_3 \; = \; & (x_1 + x_2) + a(-2x_1{x_2}^4 - 4{x_1}^2{x_2}^3 - 4{x_1}^3{x_2}^2 - 2{x_1}^4x_2) \\ & + b(-3x_1{x_2}^6 - 9{x_1}^2{x_2}^5 - 15{x_1}^3{x_2}^4 - 15{x_1}^4{x_2}^3 - 9{x_1}^5{x_2}^2 - 3{x_1}^6x_2) \qquad (4.1) \\ & + a^2(-2x_1{x_2}^8 + 8{x_1}^3{x_2}^6 + 16{x_1}^4{x_2}^5 + 16{x_1}^5{x_2}^4 + 8{x_1}^6{x_2}^3 - 2{x_1}^8x_2) + \cdots . \end{aligned}$$

As already mentioned, $E(\mathbb{Z}/_{N^s\mathbb{Z}})$ is a finite group with identity $(0 : 1 : 0)$. Since $E_1(\mathbb{Z}/_{N^s\mathbb{Z}})$ is the kernel of the reduction map from $E(\mathbb{Z}/_{N^s\mathbb{Z}})$ to $E(\mathbb{Z}/_{N\mathbb{Z}})$, we get

$$\begin{aligned} |E(\mathbb{Z}/_{N^s\mathbb{Z}})| &= |E_1(\mathbb{Z}/_{N^s\mathbb{Z}})| \cdot |E(\mathbb{Z}/_{N\mathbb{Z}})| \\ &= N^{s-1} \cdot |E(\mathbb{Z}/_{N\mathbb{Z}})|. \end{aligned}$$

For the description of the cryptosystems in the following sections we need to consider the points $P_i = (Ni : 1 : 0)$ on $E(\mathbb{Z}/N^2\mathbb{Z})$. They are an important family of points on $E(\mathbb{Z}/N^2\mathbb{Z})$. To compute $m \cdot P_i$ we have to use the above Equation 4.1 since divisions are not defined. We get that $mP_i = P_{mi}$. Similarly, when working in $E(\mathbb{Z}/N^s\mathbb{Z})$ we use the point $P_1 = (N : 1 : w(N))$, and compute mP_1 applying Equation 4.1. Furthermore, it can be shown that $N^{s-1}P_1 = (0 : 1 : 0)$.

4.2 The Elliptic Curve Paillier Scheme

In this section the elliptic curve Paillier scheme by S. Galbraith will be summarised. It is a natural generalisation of Paillier's probabilistic, homomorphic public key cryptosystem [67] to elliptic curves over rings. This scheme - as well as its generalisations that we will present in Section 4.3 and 4.4 - have exactly the properties needed for our protocols constructed in the last chapter.
See [42] for details and see Section 4.1 for some mathematical facts about elliptic curves over rings and about the p-adic theory of elliptic curves.

Key generation:
To generate a key

- compute a modulus $N = pq$ as a product of two primes $p, q > 3$.
- Choose a random elliptic curve $E : y^2z = x^3 + axz^2 + bz^3$ over $\mathbb{Z}/N\mathbb{Z}$, i. e., $\gcd(N, 6(4a^3 + 27b^2)) = 1$.
  Let $M = |E(\mathbb{F}_p)| \cdot |E(\mathbb{F}_q)|$ be the order of $E(\mathbb{Z}/N\mathbb{Z})$. Then knowledge of M is polynomial-time equivalent to knowledge of the factorisation of $N = pq$ (see [42]). Furthermore, if p, q are known then M can be computed in polynomial time using the Schoof-Atkin-Elkies algorithm (see e. g. [11]).
- Choose a point $Q = (x : y : z)$ with $\mathrm{ord}(Q)|M$ in $E(\mathbb{Z}/N^2\mathbb{Z})$. Since we have $|E(\mathbb{Z}/N^2\mathbb{Z})| = MN$ (see Section 4.1), this point can be found by taking a random point $Q' = (x' : y' : z') \in E(\mathbb{Z}/N^2\mathbb{Z})$ and setting $Q = NQ'$.
  Let $P_1 := (N : 1 : 0) \in E(\mathbb{Z}/N^2\mathbb{Z})$.

The public key consists of the modulus N (and hence the point P_1), the coefficients (a, b) of the elliptic curve, and the point Q.
The secret key is the order M of the group $E(\mathbb{Z}/N\mathbb{Z})$.

As already mentioned in Section 4.1 it holds that $mP_1 = P_m = (mN : 1 : 0)$ for $0 \leq m < N$. Since

$$(m + N)P_1 = ((m + N)N : 1 : 0) = (mN : 1 : 0) \in E(\mathbb{Z}/N^2\mathbb{Z})$$

we can also define $mP_1 = P_m$ for $m \in \mathbb{Z}/N\mathbb{Z}$. This is also valid for the generalisations given in the following sections.

Encryption:
To encrypt a message $m \in \mathbb{Z}/N\mathbb{Z}$ choose a random integer $1 \leq r < N$ and compute the point

$$C = rQ + mP_1 = rQ + P_m.$$

The ciphertext is the point $C \in E(\mathbb{Z}/N^2\mathbb{Z})$.

Decryption:
To decrypt the ciphertext C use the secret key M to compute

$$MC = r\underbrace{(MQ)}_{\mathcal{O}} + MP_m = P_{mM} = (mMN : 1 : 0).$$

Given the x-coordinate mMN interpreted in $\mathbb{Z}$ we can divide by N to obtain $mM \in \mathbb{Z}/N\mathbb{Z}$ and then multiply by the inverse of M mod N to recover the message $m \in \mathbb{Z}/N\mathbb{Z}$. (Note that we have $\mathbb{Z}/N\mathbb{Z} \simeq E_1(\mathbb{Z}/N^2\mathbb{Z})$ as mentioned in Section 4.1.)

Homomorphic Property:
This scheme is additively homomorphic since given encryptions $C_1 = r_1Q + m_1P_1$ of m_1 and $C_2 = r_2Q + m_2P_1$ of m_2 an encryption $(C_1 + C_2) = (r_1 + r_2)Q + (m_1 + m_2)P_1$ of $(m_1 + m_2)$ can be computed just by adding the ciphertexts C_1 and C_2. Hence, we can define the algorithm Add as $\mathrm{Add}(E(m_1), E(m_2)) := E(m_1) + E(m_2) = E(m_1 + m_2)$ or as

$$\mathrm{Add}(E(m_1), E(m_2)) := E(m_1) + E(m_2) + r'Q = E(m_1 + m_2)$$

with $1 \leq r' < N$ in order to blind the result. Since the message set $\mathcal{M}$ equals $\mathbb{Z}/N\mathbb{Z}$ the cryptosystem is also scalar homomorphic and the algorithm Mixed-Mult can be implemented using repeatedly the algorithm Add and some blinding algorithm as already mentioned in Chapter 2.

Remark 4.2.1. For encryption we must guarantee that the random value r is chosen from a set that is large enough, i. e., we choose $r \in \{1, \ldots, N-1\}$. When adding two ciphertexts $C_1 = r_1 Q + m_1 P_1$ and $C_2 = r_2 Q + m_2 P_1$ we might obtain a value $r_1 + r_2$ which is larger than $N - 1$. We can ignore this as long as for the so obtained new ciphertext C it holds that $C \neq (mN : 1 : 0)$, which can easily be verified. Otherwise we have to blind the messages by adding $r'Q$ where $r' \in \{1, \ldots, N-1\}$ is randomly chosen. When using the homomorphic property in the following (i. e., also in the generalisations) this check is always implicitly included.

The security analysis is very similar to that of Paillier's original scheme. Note that the elliptic curve E in this scheme is chosen randomly. Hence, it does not leak any extra information to an adversary which would help to factorise N. Here, the semantic security (as defined in [44]) in the case of passive adversaries depends on the hardness of the following problem which we call the *elliptic curve Paillier assumption*: Given a point $Q \in E(\mathbb{Z}/N^2\mathbb{Z})$ of order dividing $|E(\mathbb{Z}/N\mathbb{Z})|$ where N is the product of two large primes, and given a random point $C \in E(\mathbb{Z}/N^2\mathbb{Z})$ determine whether C lies in the subgroup generated by Q.

Note that this hardness assumption slightly differs from *Paillier's assumption*, in which he assumes the hardness of the following problem: Given $N = pq$ and a number c that is either random in $(\mathbb{Z}/N^2\mathbb{Z})^*$ or a random N'th power in $(\mathbb{Z}/N^2\mathbb{Z})^*$ decide whether c lies in the subgroup of N'th powers.

It is possible to apply standard methods to obtain more robust security properties from a semantically secure cryptosystem. For instance, in [69] a construction is described which converts Paillier's scheme so that it is semantically secure against an adaptive chosen ciphertext attack in the random oracle model.

See [42] for further security discussions or for examples.

4.3 A Generalisation of the Elliptic Curve Paillier Scheme

In a similar way as Damgård and Jurik [30] have given a generalisation of the original Paillier scheme to make it more interesting for applications Galbraith generalised the ECPS [42]. His generalisation for the elliptic curve case will be presented in this section and later on we will generalise it further to a threshold version. Furthermore, we will give a length-flexible variant of the generalised elliptic curve Paillier scheme. These generalisations use higher powers of N and have certain advantages as can be seen in [30]. So, instead of considering the ciphertext group $E(\mathbb{Z}/N^2\mathbb{Z})$ we now consider elliptic curves over $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$ for $s > 0$. In this process we have to take care of subtleties relating to the formal group, see Section 4.1, [80] and [42].

Key generation:
To generate a key

- choose a modulus $N = pq$ as a product of two primes greater than 3 and choose $s > 0$. (Thus the message set will be the group $\mathbb{Z}/N^s\mathbb{Z}$.)
- Choose a random elliptic curve $E : y^2z = x^3 + axz^2 + bz^3$ over $\mathbb{Z}/N\mathbb{Z}$, i. e., $\gcd(N, 6(4a^3 + 27b^2)) = 1$. Let $M = |E(\mathbb{F}_p)| \cdot |E(\mathbb{F}_q)|$.
- Choose a point $Q = (x : y : z)$ with $\text{ord}(Q)|M$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$. This point Q can be found by taking a random point $Q' = (x' : y' : z') \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$ and setting $Q = N^sQ'$. (We have $|E(\mathbb{Z}/N^{s+1}\mathbb{Z})| = MN^s$ (see Section 4.1).)
  Now let $P_1 := (N : 1 : w(N)) = (N : 1 : N^3 + aN^7 + \cdots) \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$. We take terms in the z-coordinate until the degree is greater than $s + 1$ (see Section 4.1). One can show that P_1 has order N^s, i. e., $N^sP_1 = \mathcal{O}$.

The public key consists of N, s, the coefficients (a, b) of the elliptic curve, and the point Q.
The corresponding secret key is the order M of the group $E(\mathbb{Z}/N\mathbb{Z})$.

Encryption:
To encrypt a message $m \in \mathbb{Z}/N^s\mathbb{Z}$ choose a random integer $1 \leq r < N^s$ and compute

the point $C = rQ + mP_1$ using Equation 4.1 of Section 4.1 (since divisions are not defined here). The ciphertext is the point $C \in E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$.

Decryption:
To recover the message $m \in \mathbb{Z}/_{N^s}\mathbb{Z}$ compute

$$MC = r(MQ) + mMP_1 = mMP_1 := m'P_1 = (m'N + \cdots : 1 : (m'N)^3 + \cdots).$$

Then $m' \in \mathbb{Z}/_{N^s}\mathbb{Z}$ can be computed iteratively and after multiplying the result by $M^{-1} \bmod N^s$ we obtain the message as $m = m'M^{-1} \bmod N^s$. Note that owing to the fact that for $s > 2$ the map ψ is not a group isomorphism but induces only the group isomorphism from $N^j(\mathbb{Z}/_{N^s}\mathbb{Z})/N^{j+1}(\mathbb{Z}/_{N^s}\mathbb{Z})$ to $E_j(\mathbb{Z}/_{N^s}\mathbb{Z})/E_{j+1}(\mathbb{Z}/_{N^s}\mathbb{Z})$, divisions are not defined here, see Section 4.1.
The iteration is as follows: We write $m' = \sum_i m'_i N^i$ in terms of its base-N representation. Let the point $m'P_1 = (x : y : z)$ be given. The x-coordinate of this point equals $\sum_i m'_i N^i \cdot N + \cdots = m'_0 N + m'_1 N^2 + \cdots + \cdots$. We can determine the value of m'_0 as $m'_0 = \frac{x}{N} \bmod N$. We can then subtract $m'_0 P_1$ from $m'P_1$ (using Equation 4.1) to obtain a new point $(x : y : z)$. From this point we can recover $m'_1 = \frac{x}{N^2} \bmod N$ and the process is iterated.

Note that for $s = 1$ we obtain the basic elliptic curve Paillier scheme.

Obviously this generalisation has the same homomorphic properties as the basic scheme. This time its semantic security is based on the assumed hardness of the following assumption, which we call *generalised elliptic curve Paillier assumption*: Given a point $Q \in E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$ of order dividing $|E(\mathbb{Z}/_N\mathbb{Z})|$ where N is the product of two large primes and given a random point $C \in E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$ determine whether C lies in the subgroup generated by Q.

4.4 A Threshold Version of the Elliptic Curve Paillier Scheme

In [30] Damgård and Jurik proposed a threshold decryption version of their generalised Paillier scheme based on an idea by Shoup [79]. This threshold version has many interesting properties and is useful for numerous applications as was shown in e. g. [30, 24, 33, 32]. In this section we adopt their approach to apply it to the generalised elliptic curve Paillier scheme. This leads to a k out of ℓ threshold decryption version of the generalised ECPS, which can be applied in a similar way to various applications, see Sections 4.6 and 4.7. Our applications are based on several protocols which we develop in Section 4.5. These protocols are the first elliptic curve versions of this kind. Furthermore, in Subsection 4.4.1 we will describe a length-flexible version of the threshold ECPS that may be useful to build e. g. mix-nets.

The idea behind threshold cryptography is to distribute the power of one authority to a group of players. For instance in a threshold decryption scheme, the secret key is shared in such a way that only a group of players is able to decrypt a ciphertext and not just one person. Here, we suggest a k out of ℓ threshold decryption scheme, which is a protocol that distributes the secret key of the generalised ECPS to a set of ℓ players so that it allows any subset of at least k out of ℓ players to decrypt a ciphertext correctly and efficiently, while it disallows any subset of less than k players to compute a decryption or to obtain any useful information. This property should also hold if an adversary corrupts some subset of less than k players working together. Obviously this should be done while keeping the homomorphic property and without degrading the security of the system.

The idea of threshold schemes goes back to Shamir who described in [78] how to share a secret via Lagrange's interpolation. It is based on the fact that a polynomial of degree $k-1$ is uniquely determined by k points. Thus given k or more points of a polynomial of degree $k-1$ it is possible to reconstruct it while less than k points do not leak any information about the polynomial.

In [79] Shoup proposed an efficient threshold variant of RSA signatures. He mentioned two main properties of such a scheme which also hold for our scheme: *Non-forgeability* means that less than k players are not able to forge a signature while *robustness* means that corrupted players are not able to prevent uncorrupted players from generating signatures. Our scheme can be proven secure in the random oracle model assuming the hardness of the generalised elliptic curve Paillier assumption. Furthermore, our threshold ECPS provides the property of Shoup's threshold sig-

nature scheme in the sense that share decryption and verification are completely non-interactive, and that the size of an individual share is independent from the number of players. These properties hold in a *static corruption model*, where an adversary chooses which players to corrupt at the very beginning of the attack and is not able to change his mind during the execution of the protocol.
The main part of Shoup's threshold RSA signature scheme is a protocol that enables a set of players to efficiently and collectively raise an input to a secret exponent d modulo an RSA modulus N, i. e., on input x each player computes a share of the result. Along with this goes a proof of correctness, i. e., a proof that the player honestly used his share in a correct way. This insures that dishonest and corrupted players can be identified by any player and observer and then disqualified. Sufficiently many correct shares of the result can be efficiently combined to compute $x^d \bmod N$. This method by Shoup was also used in [30] and can be applied to share combining in our case. Here the players multiply an input ciphertext with their secret share in such a way that these results can be combined to obtain the product MC of the ciphertext C and the secret value M. Subsequently, the remaining part of the decryption procedure can be done easily without knowledge of M. As already mentioned this method needs as a subroutine a proof of correctness. Here this proof is a proof of the equality of discrete logarithms that proves that the player behaved honestly and computed the correct value. It will be given in Section 4.5.

Our threshold scheme needs a trusted dealer in the key generation phase to set up the keys. This is a once and for all operation, i. e., the trusted dealer is not necessary for the encryption, share decryption or share combining phase. Furthermore, after distributing the keys the trusted party can delete all secret information. However, we can get along using multiparty computation techniques which allow us to do the key generation without a trusted party.

Analogously to Shoup we can summarise our model and security requirements as follows:
As *participants* in our scheme we have a set of ℓ players, a trusted dealer, and an adversary. The number of message shares needed to decrypt a ciphertext is k and the number of corrupted players is t. We require that $t \leq k - 1$ and that the adversary selects a subset of t players to corrupt from the beginning. Our scheme

can be divided into the following *phases*: In the *key generation phase* the trusted dealer generates a public key k_e along with secret key shares $s_1, \ldots, s_\ell$, and the verification keys V and $V_1, \ldots, V_\ell$. After that we can assume that the adversary knows the secret shares of the corrupted players, the public key, and the verification keys. During the *encryption phase* a message is encrypted under the public key. Then in the *share decryption phase* each player computes a message share given the ciphertext. This message share is published together with a proof of the correctness of the computation. In the *share combining phase* k message shares for which there exists a proof of correctness are combined using the public key in order to receive the desired decryption (the message) of the ciphertext. In our model the adversary is allowed to submit decryption requests to the uncorrupted players for ciphertexts of his choice. Upon such a request, the players output a message share for the given ciphertext. We say that the adversary *forges a message* if he is able to output a valid message corresponding to a ciphertext that was not submitted as a decryption request to at least $k-t$ uncorrupted players. The scheme is *non-forgeable* if it is computationally infeasible for the adversary to forge a message, i. e., if the view of the adversary that corrupts up to t players does not enable him to compute decryptions on his own. Our threshold scheme is said to be *robust* if the honest players are able to compute a valid decryption - no matter what the corrupted t players do (see e. g. [56]).

We now come to the description of our threshold ECPS, TECPS for short. The key generation of the generalised ECPS has to be extended as follows:

Key generation:

- Choose $N = pq$, where $p, q > 3$ are primes, and some $s > 0$.
- Choose a random elliptic curve $E : y^2z = x^3 + axz^2 + bz^3$ over $\mathbb{Z}/N\mathbb{Z}$, i. e., $\gcd(N, 6(4a^3 + 27b^2) = 1$. Let $M = |E(\mathbb{F}_p)| \cdot |E(\mathbb{F}_q)|$.
- Choose a point Q with $\mathrm{ord}(Q)|M$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$. Let $P_1 := (N : 1 : w(N))$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$. (For the definition of $w(N)$ see Section 4.1 and [80].)
- Pick M' to satisfy $M' = 0 \bmod M$ and $M' = 1 \bmod N^s$. (We may always

assume that $\gcd(M, N^s) = 1$.) Construct the polynomial

$$f(x) = \sum_{i=0}^{k-1} a_i x^i \in \mathbb{Z}[x]$$

by picking random $a_i \in \mathbb{Z}$ for $0 < i < k$ that are large enough[2] and $a_0 = M'$.

The secret key share of player i is $s_i = f(i)$ for $1 \leq i \leq \ell$ and the public key is $(N, s, (a, b), Q)$.
For verification of the actions of the players, we further need the following fixed public values: $V \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$, so that $\langle V \rangle = G$ and the cyclic group G should have large order, e. g. $|G| = N^s$. Furthermore, for each player we need a verification key $V_i := (\Delta s_i)V$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$, where $\Delta = \ell!$. Hence, after the key generation phase, the public key consists of $(N, s, (a, b), Q)$ and the secret key share of player i is s_i. Furthermore, V and all V_i can be seen as part of the public key as well. But note that the public values V and all V_i are only needed during the share decryption phase (see below) and not to compute an encryption. Hence they can be omitted in most of the cases and most of the protocols presented in Section 4.5.

Remark 4.4.1. Note that for any subset of k points in $\{1, \ldots, \ell\}$ the values of f at these k points uniquely determine the coefficients a_i for $0 < i < k$ of f and hence the value of f at any other point in $\{1, \ldots, \ell\}$. Furthermore, for any subset of $k - 1$ points in $\{1, \ldots, \ell\}$ the distributions of the values of f at these points are uniform and mutually independent.

The encryption phase of our threshold scheme remains the same as in the generalised ECPS:

Encryption:
To encrypt a message $m \in \mathbb{Z}/N^s\mathbb{Z}$, choose a random integer $1 \leq r < N^s$ and compute the ciphertext as $C = rQ + mP_1 \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$.

The decryption phase is divided into two steps: First *share decryption*, in which each player computes his share of the decryption using his share of the secret key.

[2] If $s, a_i \in \mathbb{Z}, f(0) = M'$ and $|M'| := K$ then it can be shown that a_i with $|a_i| = \log \ell! + K + k + \sigma$, where σ denotes the security parameter, are suitable.

Then during *share combining* the different shares of the decryption are combined to obtain the message.

Share decryption:
Given the ciphertext C, player i computes and publishes $C_i := (\Delta s_i)C$ together with a protocol run which proves that $\log_C(C_i) = \log_V(V_i)$ and that does not reveal any additional information. This convinces the other players as well as any external observer that player i has indeed multiplied by his secret share s_i. In Section 4.5 a non-interactive protocol for this task will be presented which is based on the corresponding one in [79].

For any subset S of k points in $\{1, \dots, \ell\}$ and for $i \in S$ we define

$$\lambda_{0,i}^S = \Delta \prod_{i' \in \ S \backslash \{i\}} \frac{-i'}{i - i'} \in \mathbb{Z}.$$

These values are directly derived from Lagrange's interpolation formula. Since the denominator divides $i!(\ell - i)!$ which in turn divides $\ell! = \Delta$ these values are integers. Furthermore, they can be computed efficiently. From Lagrange's interpolation formula it follows that

$$\Delta f(0) = \sum_{i \in S} \lambda_{0,i}^S f(i) = \sum_{i \in S} \lambda_{0,i}^S s_i.$$

Share combining:
Assume that we have the required k or more shares C_i together with a proof of correctness. We can then obtain the corresponding decryption by combining them, i. e., we take a subset S of k shares and combine them to

$$C' := \sum_{i \in S} \lambda_{0,i}^S C_i = \Delta \sum_{i} \lambda_{0,i}^S s_i C = \Delta^2 f(0) C = \Delta^2 M' C.$$

Since $\mathrm{ord}(Q) | M \Rightarrow \mathrm{ord}(Q) | M'$ and $N^s P_1 = (0 : 1 : 0)$ and $M' = 1 \bmod N^s$ we have

$$C' = \underbrace{\Delta^2 M' r Q}_{\mathcal{O}} + \Delta^2 M' m P_1 = \Delta^2 M' m P_1 = \Delta^2 m P_1,$$

where m is the desired plaintext. Hence we can compute m iteratively as before (see Section 4.4.1) and multiply the result by $(\Delta^2)^{-1} \bmod N^s$. Note that we can always assume that $p, q \gg \ell$.

Since the encryption procedure remains unchanged it can easily be seen that this threshold version has the same homomorphic properties as the original elliptic curve Paillier scheme.

As already mentioned, the proof of correctness of $\log_C(C_i) = \log_V(V_i)$ enables every player and observer of the scheme to verify the correct behaviour of player i and hence to disqualify him if the proof does not succeed. The proof will be described in Section 4.5. It uses a cryptographic hash function to make it non-interactive. Therefore our proof of security will be given in the random oracle model. This means that cryptographic hash functions are replaced by a random oracle. This random oracle model was informally introduced by Fiat and Shamir [40] and later formalized in Bellare and Rogaway [6]. Thereafter it was adapted in many papers. If we would do the proofs of correctness interactively instead, we could omit the random oracle. We now prove that assuming the hardness of the generalised elliptic curve Paillier assumption this threshold decryption version of the ECPS is as secure as a centralised scheme with one trusted player who performs the decryption, i. e., as secure as the generalised ECPS. This proof will be done in a static corruption model where an adversary corrupts up to $k-1$ players from the beginning. We use a standard technique to reduce the security of our threshold version to the security of its single-decryption-server counterpart, i. e., the generalised ECPS: We exhibit a simulator which has no access to any secret information but which has an oracle access to the single-decryption-server realisation of the underlying cryptosystem (see e. g. [56]). If the adversary is not able to distinguish the view of an execution of the threshold protocol and a simulated execution that uses as oracle the single-decryption-server realisation, our threshold scheme is as secure as the generalised ECPS.

Theorem 4.4.2. *Assume the random oracle model and a static adversary that corrupts up to $k-1$ players from the beginning. Furthermore, assume the hardness of the generalised elliptic curve Paillier assumption.*
Then the above scheme is correct, i. e., given any ciphertext, the decryption protocol outputs the correct plaintext, except with negligible probability.

Given an oracle that on a given ciphertext returns the corresponding plaintext, the adversary's view of the decryption protocol can be efficiently simulated with a statistically indistinguishable distribution. Hence the above threshold scheme is as secure as the generalised ECPS, robust and non-forgeable.

The proof follows very closely the corresponding proofs in [57] and [79].

Proof. If we assume that an adversary can only contribute incorrect values for the C_i's with negligible probability, correctness of the scheme follows immediately. This, in turn, is ensured by the special soundness of the protocol for the equality of discrete logarithms given for each C_i, which will be introduced and proven in Subsection 4.5.2.
Assume an oracle is given that on a given ciphertext as input returns the corresponding plaintext. To simulate the adversary's view of the decryption protocol we start from the public key $(N, s, (a, b), Q)$. Let $i_1, \ldots, i_{k-1} \in \{1, \ldots, \ell\}$ be the set of the corrupted players. We can now simulate the adversary's view by choosing the shares $s_{i_1}, \ldots, s_{i_{k-1}}$ of the corrupted players as random integers. We have already argued (see Remark 4.4.1) that these values will be statistically indistinguishable from the real values. Recall that $M' = f(0)$ and $s_i = f(i)$ and M' is fully determined by the choice of the public values N, s, and (a, b). Once M' is determined and the values $s_{i_1}, \ldots, s_{i_{k-1}}$ are chosen the shares s_i for $i \in \{1, \ldots, \ell\} \backslash \{i_1, \ldots, i_{k-1}\}$ for the uncorrupted players and the polynomial f are fixed as well. We have $f(i_1) = s_{i_1}, \ldots, f(i_{k-1}) = s_{i_{k-1}}$, but M' and f cannot be computed by the simulator. Hence, the shares $s_{i_1}, \ldots, s_{i_{k-1}}$ of the corrupted players do not leak any information about the shares of the uncorrupted players, the secret value M' or the polynomial f. The reason is that the adversary is not able to distinguish the values communicated during the protocol from random values.
Now we argue that the values V_j of the uncorrupted players and their shares C_j of a ciphertext C that can be seen during the share decryption phase do not give any extra information to an adversary. As described below these values can easily be simulated so that they fit to the above shares of the uncorrupted players: For this part of the proof we assume that the simulator has access to a decryption oracle, i. e., to a centralised decryption version - the generalised ECPS.
We now consider the verification key V as a ciphertext $V = rQ + m_0 P_1$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$

for which the simulator asks the oracle for the corresponding plaintext m_0. By doing so we can compute

$$f(0)V = M'V = M'rQ + m_0M'P_1 = m_0M'P_1 = m_0P_1.$$

Let S be the set $\{0, i_1, \dots, i_{k-1}\}$, and let

$$\lambda_{j,i}^S = \Delta \prod_{i' \in \ S \setminus \{i\}} \frac{j - i'}{i - i'}$$

be the Lagrange coefficients for interpolating the values of a polynomial in point j (times Δ) from its values in points in S, i. e., $\Delta f(j) = \sum_{i \in S} \lambda_{j,i}^S f(i)$. Then we can easily compute values V_j with $V_j = \Delta s_j V = \Delta f(j)V$ for the uncorrupted players, i. e., $j \in \{1, \dots, \ell\} \setminus \{i_1, \dots, i_{k-1}\}$, as

$$V_j = \sum_{i \in S} \lambda_{j,i}^S \cdot (f(i)V).$$

When receiving a ciphertext point C as input, we ask the oracle for the corresponding plaintext m. This gives us the possibility to compute the points

$$f(0)C = M'C = mP_1$$

which means that we can interpolate and compute the contributions $C_i = \Delta s_i C$ for the uncorrupted players in the same way as above. This argument shows why we defined the shares C_i to be $\Delta s_i C$ instead of, say, $s_i C$.
Obviously the so obtained values are correctly distributed. □

4.4.1 A Length-Flexible Variant

In [30] Damgård and Jurik proposed a length-flexible variant of their generalisation of Paillier's encryption scheme. In a similar way we show in this subsection how it is possible to adjust the block length of our threshold scheme, i. e., we propose a variant of our TECPS that can efficiently handle messages m of arbitrary length $m < N^{s'}$ where $s' \leq s$, whereas the public key and the secret key shares remain of fixed sizes.

More precisely, it is possible to decide on a value for $s' \leq s$, so that the message space $\mathcal{M}$ will be $\mathbb{Z}/_{N^{s'}}\mathbb{Z}$ with corresponding ciphertext space $\mathcal{C} = E(\mathbb{Z}/_{N^{s'+1}}\mathbb{Z})$ at any point after the keys have been generated. Even the sender can decide on the fly on a value for s' when encrypting a message. Such a length-flexible system works as follows:

The key generation can be done in the same way as above where s should be chosen large enough. Let $s' \leq s$. Note that the order M is independent of s and that for a point Q with $\operatorname{ord}(Q)|M$ in $E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$ it also holds that $\operatorname{ord}(Q)|M$ in $E(\mathbb{Z}/_{N^{s'+1}}\mathbb{Z})$. Furthermore, for $P_1 = (N : 1 : w(N)) \in E(\mathbb{Z}/_{N^{s'+1}}\mathbb{Z})$ it holds that $N^{s'}P_1 = (0 : 1 : 0)$ (see Section 4.1), and for $M' = 1 \bmod N^s$ it also holds that $M' = 1 \bmod N^{s'}$.

Encryption:
To encrypt a message m represented as a non-negative integer, choose $s' \leq s$ so that $m < N^{s'}$, choose a random integer $1 \leq r < N^{s'}$ and compute $C = rQ + mP_1$ in $E(\mathbb{Z}/_{N^{s'+1}}\mathbb{Z})$. Along with the ciphertext the value of s' is transmitted.

Share decryption and share combining work exactly the same way as before using s' instead of s.

It immediately follows that the above variant is semantically secure if the original scheme is semantically secure.

For instance, this property of length-flexibility together with the homomorphic property of the scheme is useful for election schemes as well as to build mix-nets (see Section 2.2). Mix-nets are protocols which provide anonymity for senders by collecting encrypted messages from several users. Then a collection of servers processes these, so that at the end the plaintext messages are output in randomly permuted order and sent to the intended receivers. Hence, it is not possible to trace back the messages from the receiver to the corresponding sender. The length-flexibility of the underlying encryption scheme ensures that the mix-net is able to handle messages of arbitrary size, i. e., although all messages submitted in a single run of the mix-net must have the same length in order to provide anonymity, this common length can be chosen freely for each run of the mix-net, without having to change any public information. This is especially useful for providing anonymity, e. g. for emails.

See [31] for details of mix-nets and further applications of length-flexible threshold cryptosystems to electronic voting.

4.5 Auxiliary Protocols

As mentioned above we need as a subroutine in the share decryption phase a protocol for the equality of discrete logarithms, i. e., a proof that given values $C, \tilde{C}, V, \tilde{V}$ in $E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$ where $\langle V \rangle = G$ and $|G|$ large, it holds that $\log_C(\tilde{C}) = \log_V(\tilde{V})$. Such a protocol will be presented in Subsection 4.5.2.

In [24] Cramer et al. introduced a new approach to multiparty computation based on homomorphic threshold cryptosystems. The idea of this application of homomorphic cryptosystems will be described in Section 4.6. Besides the usual properties of a threshold homomorphic cryptosystem some auxiliary protocols are needed for this approach. More precisely, we need

- a protocol for checking if a value is a valid ciphertext,
- a protocol for proving the correctness of the multiplication of an encrypted value by a constant,
- a protocol for proving the knowledge of a plaintext.

For the application of our threshold homomorphic cryptosystem to electronic voting which will be described in Section 4.7 another auxiliary protocol is needed: a protocol for proving that an encryption contains one out of two given values, without revealing which one it is. In this section we are going to develop these protocols for our homomorphic threshold cryptosystem in an analogous manner as it was done for the generalised Paillier scheme in [30, 24, 57, 32]. The special type of protocols we need are called Σ-Protocols. They will be shortly defined in the following subsection.

4.5.1 Σ-Protocols

Σ-protocols are two-party zero-knowledge protocols of a particular form. Let R be a relation consisting of pairs (x, w), where we think of x being a public instance of some computational problem, and w being a *witness*, i. e., a solution to that instance. We will be concerned with protocols of the following form: This protocol gets x as a common input for the prover $\mathcal{P}$ and the verifier $\mathcal{V}$, and the prover additionally knows a witness w as a private input such that $(x, w) \in R$.

1. $\mathcal{P}$ sends a message a to $\mathcal{V}$.
2. $\mathcal{V}$ sends a random t-bit challenge e to $\mathcal{P}$.
3. $\mathcal{P}$ sends a reply z.
4. $\mathcal{V}$ decides to accept or reject based on the data he has seen, i. e., x, a, e, z.

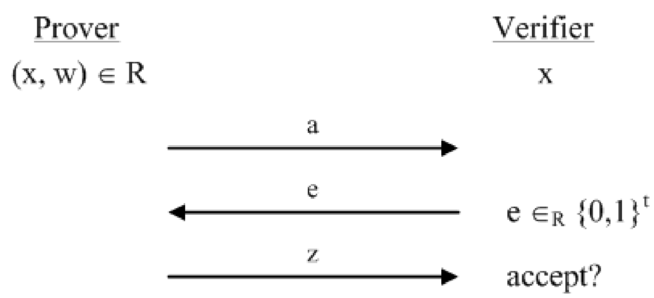


Figure 4.1: Σ-Protocol

We will assume that both $\mathcal{P}$ and $\mathcal{V}$ are probabilistic, polynomial time machines. Hence, $\mathcal{P}$'s only advantage over $\mathcal{V}$ is the knowledge of the witness w.

Definition 4.5.1. A protocol is said to be a Σ-*protocol* for relation R if we have the following:

- The protocol is of the above form.
- *Completeness*: If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol, the verifier always accepts.
- There exists a polynomial-time simulator, which on input x and a random challenge e outputs an accepting conversation of the form (a, e, z) with the same probability distribution as conversations between honest $\mathcal{P}$ and $\mathcal{V}$ on input x and challenge e. This property is called *special honest-verifier zero-knowledge*.

- *Special soundness*: A cheating prover can answer only one of the possible challenges. More precisely, from any common input x and any pair of accepting conversations (a, e, z) and (a, e', z') where $e \neq e'$, we can efficiently compute w so that $(x, w) \in R$.

Note that the notion honest-verifier zero-knowledge proof can also be found in the literature instead of Σ-protocols.
It is easy to verify that the properties of Σ-protocols are invariant under parallel composition. For more details and properties of Σ-protocols see [29] and [24].

4.5.2 Protocol for the Equality of Discrete Logarithms

For the share decryption phase we need a subroutine proving that given values $C, \tilde{C}, V, \tilde{V}$ in $E(\mathbb{Z}/N^{s+1}\mathbb{Z})$ where $\langle V \rangle = G$ and $|G|$ is large, it holds that

$$\log_C(\tilde{C}) = \log_V(\tilde{V}).$$

With such a protocol run coming together with the published result $\tilde{C}$ (the ciphertext share C_i) of a player P_i, the other players (and any observer) of our threshold decryption scheme will be convinced of the correctness of his computation. Thus here the player P_i is the prover. The values $C, \tilde{C}, V, \tilde{V}$ as well as the public key are public values in our protocol. Player P_i knows additionally $\Delta s_i =: y$ which is his secret input.
We need a Σ-protocol for the relation

$$R = \{((C, \tilde{C}, V, \tilde{V}), y) \mid C, \tilde{C}, V, \tilde{V} \in E(\mathbb{Z}/N^{s+1}\mathbb{Z}), y = \log_C(\tilde{C}) = \log_V(\tilde{V})\}.$$

It works as follows:

Protocol for the equality of discrete logarithms
Input: $k_e = (N, s, (a, b), Q)$, $C, \tilde{C}, V, \tilde{V} \in (E(\mathbb{Z}/N^{s+1}\mathbb{Z}), +)$, where $\langle V \rangle = G$ and $|G|$ is large
Private input for the prover: y such that $y = \log_C(\tilde{C}) = \log_V(\tilde{V})$.
Note that in our application the length of y will be bounded by $s|N||M|$ bit since

$V \in E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$. Here, $|N|$ and $|M|$ denote the bit length of N and M, respectively. W.l.o.g. these bit lengths are known as well.

1. The prover $\mathcal{P}$ chooses a random number r of bit length $s|N||M| + 2t$, where t is a secondary security parameter. He computes $A = rC$ and $B = rV$ in $E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$ and sends (A, B) to the verifier $\mathcal{V}$.
2. The verifier chooses a random t-bit challenge e which he sends to the prover.
3. $\mathcal{P}$ computes the number $z = r + ey$ and sends it to $\mathcal{V}$.
4. The verifier accepts the proof if and only if $zC = A + e\tilde{C}$ and $zV = B + e\tilde{V}$ in $E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$.

Lemma 4.5.2. The above protocol for the equality of discrete logarithms is a Σ-protocol.

Proof. The completeness of this protocol under an honest prover is obvious. Special honest-verifier zero-knowledge holds because, for a random t-bit number e and a random $s|N||M| + 2t$-bit number z we have that $((zC - e\tilde{C}, zV - e\tilde{V}), e, z)$ is an accepting conversation with the same probability distribution. For special soundness we assume that $\mathcal{P}$ does not know y. If the prover can send correct responses z and z' to two different challenges e and e' that satisfy the verifier's check where w.l.o.g. $e > e'$, i. e.,

$$\begin{aligned} zC &= A + e\tilde{C} \quad &\text{and} \quad zV &= B + e\tilde{V}, \\ z'C &= A + e'\tilde{C} \quad &\text{and} \quad z'V &= B + e'\tilde{V} \end{aligned}$$

then

$$(z - z')C = (A + e\tilde{C}) - (A + e'\tilde{C}) = (e - e')\tilde{C}$$

and

$$(z - z')V = (B + e\tilde{V}) - (B + e'\tilde{V}) = (e - e')\tilde{V},$$

and hence

$$\log_C(\tilde{C}) = \log_V(\tilde{V}) = (z - z')(e - e')^{-1}$$

the discrete logarithms is efficiently computable. This contradiction shows that the prover is able to answer at most one challenge e correctly. Therefore the probability of acceptance of a dishonest prover is 2^{-t} which is negligible in t. $\square$

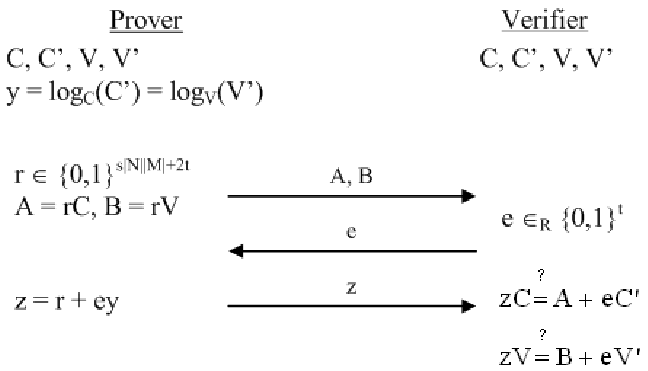


Figure 4.2: Protocol for the Equality of Discrete Logarithms

Using the Fiat-Shamir heuristic [40] which requires a hash function this protocol can be made non-interactive, where security can be proven in the random oracle model:

Let $\mathcal{H}$ be a hash function, whose output is a t-bit integer ($t = 128$, say).

1. The prover $\mathcal{P}$ chooses a random number r of bit length $s|N||M| + 2t$. He computes $A = rC$ and $B = rV$ in $E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$, sets $e = \mathcal{H}(A, B, C, \tilde{C})$ and computes $z = r + ey$. He defines the proof of correctness to be (e, z) and sends this to the verifier $\mathcal{V}$.

2. To verify this proof $\mathcal{V}$ checks that $e = \mathcal{H}(zC - e\tilde{C}, zV - e\tilde{V}, C, \tilde{C})$.

In this non-interactive version of the protocol we can exclude $V, \tilde{V}$ in the input to $\mathcal{H}$ since in our application, they are fixed and chosen by a honest dealer as being the verification keys. If we assume the prover's claim to be true, i. e., the equality of the discrete logarithms holds then the correctness of this protocol follows immediately. Assuming the random oracle model, i. e., replacing $\mathcal{H}$ by a random function, we can show soundness and special honest-verifier zero-knowledge as above.

The Fiat-Shamir heuristic can also be used to make all of the following protocols of this section non-interactive.

4.5.3 Check of Ciphertextness

Now we will describe a protocol that given a value C and the public key k_e checks whether C is a valid ciphertext. For the parameters of our threshold scheme that means that given a point C and the public key $k_e = (N, s, (a, b), Q)$ check whether there exists a message $m \in \mathbb{Z}/N^s\mathbb{Z}$ and a random element $0 \leq r < N^s$ such that $C = E_{k_e,r}(m) = rQ + mP_1 \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$. Note that the values V and all V_i are only needed for the share decryption phase in our threshold scheme. They can be omitted as public inputs in all our following protocols.
The property $C = rQ + mP_1$ is obviously equivalent to $C - mP_1 = rQ$, i.e., to $C - mP_1$ being an encryption $E_{k_e,r}(0)$ of 0 under the public key k_e and a random number r.
Hence, we would like to construct a Σ-protocol for the relation

$$R = \{((k_e, C), r) \mid C = E_{k_e,r}(0) \text{ where } k_e \text{ is the public key}\}.$$

A protocol for this can easily be obtained from the corresponding one in [57]:

Protocol for checking whether an element is an encryption of 0
Input: $k_e = (N, S, (a, b), Q), C$
Private input for the prover: r with $0 \leq r < N^s$ so that $C = rQ = E_{k_e,r}(0)$.

1. The prover $\mathcal{P}$ chooses r' with $0 \leq r' < N^s$ at random. He computes

$$A = r'Q = E_{k_e,r'}(0)$$

   and sends A to the verifier $\mathcal{V}$.

2. $\mathcal{V}$ challenges $\mathcal{P}$ with a random t-bit string e.

3. $\mathcal{P}$ computes $z = r' + er$ and sends z to $\mathcal{V}$.

4. $\mathcal{V}$ checks whether $A, C \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$ and that $E_{k_e,z}(0) = A + eC$. He accepts if and only if this is the case.

Lemma 4.5.3. The above protocol for checking whether an element is an encryption of 0 is a Σ-protocol.

Proof. For completeness we have to show that the verifier always accepts if the prover and the verifier follow the protocol. This holds as follows:

$$A + eC = r'Q + eC = r'Q + erQ = (r' + er)Q = zQ = E_{k_e,z}(0).$$

For special honest verifier zero-knowledge we have to simulate a conversation (A, e, z) with the same probability distribution. Therefore the simulator chooses z and e randomly in their respective domains. He sets $A = E_{k_e,z}(0) - eC$ and outputs (A, e, z), which is obviously a conversation with the same probability distribution. For special soundness we assume that there are two accepting conversations (A, e, z) and (A, e', z') where $e \neq e'$ and w.l.o.g. $e > e'$. Hence, we obtain

$$E_{k_e,z}(0) = A + eC \text{ and } E_{k_e,z'}(0) = A + e'C.$$

Therefore

$$(e - e')C = E_{k_e,(z-z')}(0) = (z - z')Q.$$

Since $e \neq e'$ we can set $v := (z - z')(e - e')^{-1}$. So,

$$C = (z - z')(e - e')^{-1}Q = vQ = E_{k_e,v}(0),$$

i. e., the necessary values for an encryption of 0 can indeed be efficiently computed. □

In our application of this and the following protocols we assume that the modulus $N = pq$ has two prime factors of roughly the same size. Hence, if $|N|$ is the bit length of N, we can set $t = |N|/2$ to ensure that the probability of a cheating prover to make the verifier accept is $\leq 2^{-t}$.

4.5.4 Proof of Correct Multiplication

We now describe a protocol for proving that a given ciphertext is the product of an encrypted value by a plaintext. Hence, we define the relation

$$R = \{((C_{m_1}, C_{m_2}, D), (m_2, r_2, r_3)) \mid C_{m_1} = E(m_1), C_{m_2} = E_{r_2}(m_2), D = m_2 C_{m_1} + r_3 Q\},$$

where all encryptions are done under the same public key $k_e = (N, s, (a, b), Q)$. Using the algorithm Mixed-Mult to express the relation this means that

$$D = \text{Mixed-Mult}(m_2, C_{m_1}),$$

i. e., D is an encryption of $m_2 m_1$. A protocol for this relation is similar to the one proposed in [24] for the threshold Paillier scheme:

Protocol for proving correct multiplication
Input: $k_e = (N, s, (a, b), Q)$ and C_{m_1}, C_{m_2}, D, so that

$$C_{m_1} = r_1 Q + m_1 P_1,\ C_{m_2} = r_2 Q + m_2 P_1,\ D = m_2 C_{m_1} + r_3 Q$$

Private input for the prover: m_2 and r_2, r_3

1. The prover $\mathcal{P}$ chooses $m \in \mathbb{Z}/_{N^s}\mathbb{Z}$ and $0 \leq v, u < N^s$ at random. He computes

   $$A = mC_{m_1} + vQ \text{ and } B = mP_1 + uQ.$$

   $\mathcal{P}$ sends (A, B) to the verifier $\mathcal{V}$.

2. $\mathcal{V}$ sends a random t-bit challenge e to $\mathcal{P}$.

3. The prover $\mathcal{P}$ computes

   $$w = m + em_2,\ y = v + er_3,\ z = u + er_2$$

   and sends (w, y, z) to $\mathcal{V}$.

4. The verifier checks that

   $$wP_1 + zQ = B + eC_{m_2},\ wC_{m_1} + yQ = A + eD.$$

   He accepts if and only if this is the case.

Lemma 4.5.4. The above protocol for proving correct multiplication is a Σ-protocol proving knowledge of m_2, r_2, r_3 such that $C_{m_2} = r_2 Q + m_2 P_1$ and $D = m_2 C_{m_1} + r_3 Q$, i. e., proving that D encrypts $m_2 m_1$.

Proof. Showing completeness is straightforward since

$$wP_1 + zQ = (m + em_2)P_1 + (u + er_2)Q = mP_1 + uQ + e(r_2Q + m_2P_1) = B + eC_{m_2}$$

and

$$wC_{m_1} + yQ = (m + em_2)C_{m_1} + (v + er_3)Q = mC_{m_1} + vQ + e(m_2C_{m_1} + r_3Q) = A + eD.$$

For special honest verifier zero-knowledge, given any challenge e a correctly distributed conversation can be simulated as follows: the simulator chooses the values w, y, z at random in their respective domains. He then computes matching values A, B using the equations $wP_1 + zQ = B + eC_{m_2}$ and $wC_{m_1} + yQ = A + eD$.
For soundness we assume that for some value of (A, B) the prover $\mathcal{P}$ can correctly answer two different challenges e, e' where w.l.o.g. $e > e'$. Hence, the following equations are satisfied

$$wP_1 + zQ = B + eC_{m_2},\ wC_{m_1} + yQ = A + eD$$

and

$$w'P_1 + z'Q = B + e'C_{m_2},\ w'C_{m_1} + y'Q = A + e'D.$$

This implies that

$$(w - w')P_1 + (z - z')Q = (e - e')C_{m_2}$$

and

$$(w - w')C_{m_1} + (y - y')Q = (e - e')D.$$

Hence

$$C_{m_2} = (w - w')(e - e')^{-1}P_1 + (z - z')(e - e')^{-1}Q$$

and

$$D = (w - w')(e - e')^{-1}C_{m_1} + (y - y')(e - e')^{-1}Q.$$

We can conclude that $m_2 = (w - w')(e - e')^{-1}, r_2 = (z - z')(e - e')^{-1}$ and that $r_3 = (y - y')(e - e')^{-1}$. Thus D is indeed an encryption of m_2m_1. □

4.5.5 Proof of Plaintext Knowledge

We want to obtain a Σ-protocol that given an encryption $C = E_{k_e}(m)$ of m proves that $\mathcal{P}$ knows the corresponding message m. Such a protocol is contained implicitly in the above protocol for proving correct multiplication. Here, the relation R is obviously

$$R = \{(k_e, C_{m_2}), (m_2, r_2)) \mid m_2 \in \mathcal{M}, C_{m_2} = E_{k_e, r_2}(m_2)\}.$$

Protocol for proving plaintext knowledge
Input: $k_e = (N, S, (a, b), Q)$ and $C_{m_2} = r_2 Q + m_2 P_1$
Private input for the prover: m_2 and r_2

1. $\mathcal{P}$ chooses $m \in \mathbb{Z}/_{N^s}\mathbb{Z}$ and $0 \leq u < N^s$ at random. He computes $B = mP_1 + uQ$ and sends B to $\mathcal{V}$.
2. $\mathcal{V}$ sends a random t-bit challenge e to $\mathcal{P}$.
3. $\mathcal{P}$ computes $w = m + em_2,\ z = u + er_2$ and sends (w, z) to $\mathcal{V}$.
4. $\mathcal{V}$ checks that $wP_1 + zQ = B + eC_{m_2}$ and accepts if and only if this is the case.

Lemma 4.5.5. The above protocol for proving plaintext knowledge is a Σ-protocol proving knowledge of m_2 and r_2 such that $C_{m_2} = r_2 Q + m_2 P_1$.

The proof follows immediately from Lemma 4.5.4.

4.5.6 1-out-of-2 Protocol

We now describe a Σ-protocol that given two ciphertexts C_1, C_2, and the public key k_e proves that the prover knows one of the corresponding messages without revealing which one it is. This property is called *witness indistinguishability*. As before this is equivalent to proving that the prover knows one corresponding random value r to two encryptions of 0, i. e., if $C_1 = E_{k_e, r_1}(0) = r_1 Q$ and $C_2 = E_{k_e, r_2}(0) = r_2 Q$ are encryptions under the same public key k_e and the random number r_1 and r_2, respectively, to show that $\mathcal{P}$ knows either r_1 or r_2 without revealing either of the values. Due to Lemma 4.5.3 and using the techniques from [25] we can immediately build such a protocol. We will assume without loss of generality that the prover knows r_1 such that $C_1 = E_{k_e,r_1}(0)$ and where $\mathcal{S}$ denotes the honest-verifier simulator for the protocol for checking whether an element is an encryption of 0 of Subsection 4.5.3.

1-out-of-2 protocol
Input: $k_e = (N, S, (a, b), Q), C_1, C_2$
Private input for the prover: r_1 with $0 \leq r_1 < N^s$ so that $C_1 = r_1 Q = E_{k_e,r_1}(0)$.

1. The prover $\mathcal{P}$ chooses randomly r with $0 \leq r < N^s$. He invokes the honest-verifier simulator $\mathcal{S}$ on inputs k_e, C_2 to obtain a conversation (A_2, e_2, z_2). $\mathcal{P}$ computes $A_1 = rQ = E_{k_e,r}(0)$ and sends A_1 and A_2 to the verifier $\mathcal{V}$.

2. $\mathcal{V}$ chooses randomly a t-bit challenge e and sends it to $\mathcal{P}$.

3. $\mathcal{P}$ computes $e_1 = e - e_2$, where w.l.o.g. $e > e_2$, and $z_1 = r + e_1 r_1$. He sends e_1, z_1, e_2, z_2 to $\mathcal{V}$.

4. $\mathcal{V}$ checks whether $A_1, A_2, C_1, C_2 \in E(\mathbb{Z}/N^{s+1}\mathbb{Z})$ and that $E_{k_e,z_1}(0) = A_1 + e_1 C_1$ and $E_{k_e,z_2}(0) = A_2 + e_2 C_2$. He accepts if and only if this is the case.

Now Lemma 4.5.3 and the proof techniques from [25] imply the following Lemma.

Lemma 4.5.6. The above 1-out-of-2 protocol is a Σ-protocol.

4.6 Application to Multiparty Computation

Given a semantically secure threshold homomorphic cryptosystem and some auxiliary protocols it is possible to obtain general multiparty computation protocols which are secure against an active and static adversary that corrupts any minority of the players, see [24]. For our threshold version of the generalised elliptic curve Paillier scheme we constructed the auxiliary protocols that are needed (see Section 4.5). Hence, the proposed scheme can be used to build such multiparty computation

protocols. In this section we will summarise the results and ideas of the approach by Cramer et al. [24].
The idea of multiparty computation dates back to the papers by Goldreich, Micali, Wigderson [46] and by Yao [84]. They were able to prove that n players can efficiently compute an agreed n-input function of their inputs in a secure way. In such a multiparty computation protocol security means that everyone learns the correct output while the privacy of each player's input is guaranteed. Note that the scenarios considered in Chapter 3 belong to the same area. There are several different types of adversaries possible for such kind of problem, and we can distinguish between active and passive corruption. If an adversary can corrupt a set of players to make them behave the way he wants, we say that the adversary is *active*, else if the adversary obtains "only" the complete information held by the corrupted players while the players still are able to execute the protocol correctly, he is called *passive*. If the set of corrupted players is fixed from the beginning, the adversary is called *static*. If the adversary can at any time during the protocol choose to corrupt a new player based on all information he has at this time, he is called *adaptive*. The protocols by Goldreich et al. and Yao can be proven secure against an active, static, polynomial time bounded adversary who can corrupt any set of at most $n/2 - 1$ players.
Several protocols for multiparty computations have been proposed over the years to improve efficiency, see e. g. [23]. Most of the proposed protocols have been based on verifiable secret sharing schemes. A *verifiable secret sharing scheme* allows a dealer to securely distribute a secret s among a set of players, where the dealer and/or some players may be cheating. Here it is guaranteed that if the dealer is honest, then the cheaters obtain no information about s, and all honest players are able to reconstruct the secret s even against the actions of the cheating players.
The proposal by Cramer et al. basing multiparty computation protocols on a threshold homomorphic cryptosystem instead of secret sharing schemes leads to more efficient protocols since it reduces the total number of bits that have to be sent to $O(n\sigma|C|)$, where n is the number of players, σ is the security parameter and $|C|$ the size of the circuit C computing the desired function. All previous protocols which were secure against active adversaries required $\Omega(n^2\sigma|C|)$ bits to broadcast.

The idea of Cramer et al. is basically as follows: given a semantically secure public-key cryptosystem with threshold decryption where the message space $\mathcal{M}$ is assumed to be a ring. (In our case we have $\mathcal{M} = \mathbb{Z}/_{N^s}\mathbb{Z}$). The following homomorphic properties are required: given encryptions $E(a)$ and $E(b)$ of a and b under the same public key we can efficiently compute an encryption of the sum $E(a+b)$. Furthermore, from an encryption $E(b)$ and a plaintext $a \in \mathcal{M}$ it should be easy to compute a random encryption of $E(ab)$. Obviously these properties are just the additively homomorphic property and the scalar-multiplicativity which hold for our threshold decryption ECPS.

Finally, some secure sub-protocols must be available: a protocol for checking if a value is a valid ciphertext, a protocol for proving the correctness of the multiplication of an encrypted value by a constant, and a protocol for proving the knowledge of a plaintext. Based on our scheme these protocols were described in Section 4.5.

To securely compute an n-input function we consider a circuit with addition and multiplication gates computing the function and evaluate the circuit gate by gate.

To start the multiparty computation protocol each player $\mathcal{P}_i$, for $1 \le i \le n$, publishes an encryption $E(x_i)$ of his private input x_i together with a Σ-protocol proving his knowledge of x_i, see Subsection 4.5.3. Any operation involving addition or multiplication by a constant can then be performed without further interaction using the algorithms Add and Mixed-Mult of the encryption scheme. Hence, we just need a protocol for securely computing an encryption of the product $E(c) = E(a \cdot b)$ given encryptions of $E(a)$ and $E(b)$ but not the values a and b themselves. This in turn can be done by the following protocol:

Multiparty computation protocol for multiplication of ciphertexts

Input: $E(a), E(b)$
Output: $E(c) = E(a \cdot b)$

1. Each player $\mathcal{P}_i$ chooses a random value $r_i \in \mathcal{M}$ and broadcasts an encryption $E(r_i)$ of it. Furthermore, all players prove that they know their value r_i using the corresponding sub-protocol.

2. Let $r = r_1 + \cdots + r_n$. All players can now compute an encryption $E(a+r)$ by applying iteratively the algorithm Add on $E(a)$ and $E(r_1), \ldots, E(r_n)$. The so

obtained ciphertext $E(a+r)$ is then decrypted using the protocol for threshold decryption. Now all players know $a + r$.

3. Player $\mathcal{P}_1$ sets $a_1 = (a + r) - r_1$ while all other players $\mathcal{P}_i$ for $1 < i \leq n$ set $a_i = -r_i$. Note that every player is able to compute an encryption of each a_i and that we have $a = a_1 + \cdots + a_n$.

4. Each player $\mathcal{P}_i$ computes an encryption $E(a_i b) = \text{Mixed-Mult}(a_i, E(b))$ and broadcasts it together with a proof for the correctness of the multiplication using the corresponding sub-protocol from Subsection 4.5.4 on inputs $E(b), E(a_i)$ and $E(a_i b)$.

5. Let S be the set of players for which the previous step succeeded, and let S^c be the complement of S. We now first compute $E(\sum_{i\in S^c} a_i)$ using the algorithm Add. Then using threshold decryption we decrypt this ciphertext $E(\sum_{i\in S^c} a_i)$ which gives us the value $a_{S^c} := \sum_{i\in S^c} a_i$. This allows everyone to compute an encryption $E(a_{S^c} b)$ by using the algorithm Mixed-Mult. From this and the encryptions $E(a_i b)$ for $i \in S$, all players are able to compute an encryption $E(\sum_{i\in S} a_i b + a_{S^c} b)$, which is indeed an encryption of ab.

At the final state the known encryptions of the output values can be decrypted using threshold decryption to obtain the function value. Intuitively this is secure if the underlying cryptosystem is secure, since (apart from the output) all values that are decrypted are either random values or values already known to the adversary. The efficiency of this protocol depends entirely on the efficiency of the underlying threshold decryption scheme and the corresponding sub-protocols.

For a formal description, a proof of security and details see [24].

4.7 Application to Electronic Voting

In this section we will describe how our homomorphic threshold cryptosystem can be used to obtain a multi-authority secret-ballot election scheme that guarantees robustness, universal verifiability and computational privacy. Our construction is based on an idea of Cramer, Gennaro and Schoenmakers [27] who proposed such an election scheme based on a threshold version of the ElGamal scheme [37] (see Section 3.4.1). We build our voting protocol in an analogous way as the Damgård et al. protocol presented in [32].

The scheme described by Cramer et al. is based on a model by Benaloh et al. [21, 9, 7] where the active parties are divided into a set of voters $\mathcal{V}_1, \ldots, \mathcal{V}_m$ and a set of tallying authorities $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Furthermore, a so-called *bulletin board* B is needed which is accessible by all parties even passive observers to achieve universal verifiability. Such a bulletin board is like a public broadcast channel with memory. It has the property that any active or passive party is able to read the contents of it. Furthermore, there is a designated section for each active participant where he can post his messages. Hence we can identify which player each message comes from. Additionally it is not possible to delete any information from the bulletin board. For instance, such a bulletin board can be implemented in a secure way by using an already deployed public key infrastructure and a server replication technique to prevent denial of service attacks.

In our voting protocol a voter simply posts a particular encryption under the TECPS of the vote to the bulletin board accompanied by a proof of validity that shows that the ballot indeed contains a valid vote. Here the ballot is encrypted with the public key of the authorities. Since the encryption method used is additively homomorphic, the final tally can be obtained as "sum" of all votes. Furthermore, it is verifiable by any observer of the election against the "sum" of all submitted ballots. Due to the properties of our threshold decryption scheme the benign or malign failure of some tallying authorities can be tolerated while the privacy of the individual votes will be guaranteed, i. e., the correctness of the decryption will be assured even in the presence of malicious authorities which ensures universal verifiability. In the proposed protocol the private key is never reconstructed, and only used implicitly when the authorities cooperate to decrypt the final tally.

We assume that the purpose of the election is to elect a winner among L candidates, and that each voter is allowed to vote for $t < L$ candidates.

Properties of elections

There are many properties of election schemes that should be considered. We list the properties that were considered in the approach by [27]. These properties also hold for our protocol.

Eligibility: Only eligible voters are able to vote and each eligible voter can cast a single vote.

Since we assume that a bulletin board is available this property follows immediately.

Universal Verifiability: The fairness of the election can be checked by any party including passive observers. That means that any party is able to check that the published final tally is consistent with the correctly cast ballots. This property also includes that any party can check whether ballots are correctly cast, and that only invalid ballots are discarded.

Privacy: Privacy of an individual vote is assured against any reasonably sized coalition of participants (excluding the voter himself), i. e., unless the number of colluding parties exceeds our threshold k, different ballots are indistinguishable irrespective of the contained votes. *Information-theoretic* privacy is achieved when the ballots are indistinguishable independent of any cryptographic assumption; otherwise *computational* privacy is achieved.

Robustness: The faulty behaviour either benign or malicious of any reasonably sized coalition of parties can be tolerated. This includes the property that no coalition of voters of any size can disrupt a large-scale election, i. e., we can detect and discard any cheating voter.

No vote duplication: It should be impossible to copy another voter's vote even without knowing what the copied vote is.

No interaction between voters: During the execution of the voting protocol the voters do not have to interact with each other.

As mentioned in Section 4.5 we use the Fiat-Shamir heuristic which enables us to make our proofs non-interactively. The hash function needed will be denoted by h. Since we base the voting protocol on the TECPS, we further assume that an instance of the threshold ECPS with public key $(N, s, (a, b), Q)$ has been set up. The

tallying authorities $\mathcal{A}_i$ for $1 \leq i \leq n$ will act as decryption servers. To guarantee the correctness of the final tally we also have to assume that $N^s > m^L$, where m is the number of voters and L the number of candidates. Note that this can always be fulfilled by choosing s or N large enough.
In the following voting protocol for two candidates we will use the notation $Proof_{\mathcal{V}_i}(S)$ as introduced in [32]. Here S denotes a logical statement which indicates which protocol from Section 4.5 will be used. $Proof_{\mathcal{V}_i}(S) = (e, z)$ will be a bit string created by voter $\mathcal{V}_i$. $\mathcal{V}_i$ uses the appropriate protocol that can be used to interactively prove S. He computes the first message a in this protocol, computes $e = h(a, S, ID(\mathcal{V}_i))$ where $ID(\mathcal{V}_i)$ is a unique public string identifying $\mathcal{V}_i$. He takes the result e of this as the challenge from the verifier and computes the corresponding answer z. Note that we have to include $ID(\mathcal{V}_i)$ in the input of h to prevent vote duplication. Furthermore, note that in all auxiliary protocols it can always be easily computed what a should have been given S, z, e, had the proof been correct. This is done by just plugging z and e into the verifiers check. Thus, such a proof can be checked efficiently by checking whether $e = h(a, S, ID(\mathcal{V}_i))$.
Now we are able to explain a protocol for the case $L = 2$, i. e., a protocol in which the voter votes for one out of two candidates. Since this is equivalent to a yes/no vote, each vote can be considered as '0' for no and '1' for yes.

Voting protocol for two candidates

1. Each voter $\mathcal{V}_i$ decides on his vote v_i. He computes $C_i = E(v_i, r_i)$, where r_i is chosen at random. He creates $Proof_{\mathcal{V}_i}(C_i$ or $C_i - P_1$ is an encryption of 0) based on the 1-out-of-2 protocol of Subsection 4.5.6. He posts his encrypted vote together with the proof to the bulletin board B.

2. Each tally authority A_j does the following:

   - Set $C = \mathcal{O}$.
   - For all i: check the proof written by $\mathcal{V}_i$ on B. If it is valid, then set $C := C + C_i \in E(\mathbb{Z}/_{N^{s+1}}\mathbb{Z})$.
   - $\mathcal{A}_j$ uses C as the input ciphertext and executes his part of the threshold decryption protocol. He posts his result to the bulletin board B.

3. After discarding invalid messages, the combining of these decryption shares written by the A_j's can now be done by anyone, i. e., the reconstruction of the plaintext corresponding to E. For simplicity we assume now that all votes are valid. Then we obtain

$$C = \sum_i E(v_i, r_i) = E(\sum_i v_i \bmod N^s, \sum r_i \bmod N^s).$$

Hence, the result of the decryption, i. e., the final tally is $\sum_i v_i \bmod N^s$ which is $\sum_i v_i$ since we have $N^s > m$.

Analogously to [27] we can now prove the following:

Lemma 4.7.1. Assuming the hardness of the generalised elliptic curve Paillier assumption, our election scheme provides universal verifiability, computational privacy, robustness and prevents vote duplication. Furthermore, in the random oracle model no interaction between voters is needed.

Proof. Since the proofs of validity for the ballots C_i are made non-interactively, they are verifiable by any observer. Furthermore, any observer can check the final tally with respect to all valid ballots. This gives us universal verifiability.
Assuming the hardness of the generalised elliptic curve Paillier assumption the k out of ℓ TECPS is semantically secure and hence no information about the vote is leaked given $E(v_i, r_i)$. If we assume that at most $k-1$ tallying authorities are corrupted, we obtain computational privacy of the individual votes.
Note that the proof of validity cast with each vote is witness indistinguishable, i. e., it gives no hint on the corresponding witness used in the proof. Thus it does not help to break privacy.
Special soundness of the proofs of validity guarantees that a voter cannot cast bogus ballots. This ensures robustness with respect to malicious voters. Robustness with respect to at most $n-k$ malicious authorities is inherited from the robustness of the key generation and decryption protocols.
The hash function h needed for the proofs of validity gets the user identity as input. This prevents vote duplication. □

Extension to Multi-Candidate Elections

Instead of making a choice between two candidates it is often required to choose one or more out of several candidates. There are several ways to tackle this problem, i. e., to generalise the above approach to $L > 2$ candidates. A very simple way as described in [32] is to make L parallel yes/no votes as above. A voter votes 1 for the candidates he wants and 0 for the others. Hence, each voter $\mathcal{V}_i$ sends L votes in parallel which are of the following form:

$$C_{ij} = E(v_{ij}, r_{ij})$$

where $0 \leq j \leq L-1$. Here, r_{ij} are chosen at random and v_{ij} is $\mathcal{V}_i$'s vote for candidate number j. He also creates $Proof_{\mathcal{V}_i}(C_{ij}$ or $C_{ij} - P_1$ is an encryption of 0) based on the 1-out-of-2 protocol of Subsection 4.5.6. He then computes $\sum_{j=0}^{L-1} r_{ij} \bmod N^s$ and posts it to the bulletin board B. This proves that he voted for exactly t out of L candidates since any observer can check that $\sum_{j=0}^{L-1} E(v_{ij}, r_{ij})$ is an encryption of t. Due to the proof of validity we know that all individual votes are either 0 or 1 which shows that this check is sufficient. Now by decryption of the L values C_{ij} the number of votes each candidates received is obtained.

Note that the complexity of the described protocol is as well linear in the number of voters as in the number of candidates.

4.8 Application to Commitment Schemes

For a commitment scheme a new and very strong security notion called *universal composability* was introduced by Canetti and Fischlin [17]. This notion was adopted by Damgård and Nielsen [33] who presented a new construction of universally composable commitment schemes based on so called *special mixed commitment schemes.* Furthermore, they have shown that it is possible to base such a special mixed commitment scheme on the Paillier scheme. Using their results we show that it is possible to base a special mixed commitment scheme on a slightly modified version of the ECPS, and thus on the elliptic curve Paillier assumption instead of Paillier's

original assumption. The construction of Damgård and Nielsen is based on q one-way homomorphisms which will be introduced in Subsection 4.8.2. Furthermore, they prove that for the so obtain special mixed commitment scheme it is possible to build protocols for proving Boolean relations between committed values defined by any Boolean function as well as protocols for proving additive and multiplicative relations.
After explaining the idea behind commitment schemes and explaining their main properties we prove that it is possible to modify the ECPS in such a way that it fulfills the requirements of a q one-way homomorphism. Hence, using the results of [33] we obtain a special mixed commitment scheme based on the elliptic curve Paillier assumption with exactly the same properties.

Commitment schemes are a fundamental primitive in both theory and practice of cryptography, e. g. for zero-knowledge proofs (e. g., [47, 14, 28]), general function evaluation protocols (e. g. [46, 43]), contract-signing, electronic commerce or coin flipping. Informally in a commitment scheme a player $\mathcal{P}$ is able to commit to a self-chosen value of a finite set in such a way that he cannot change his mind later on. Furthermore, he does not have to reveal his choice to other players $\mathcal{V}$ at this time, i. e., to reveal to which value he is committing to. He may to choose to do so at some later time though.

As an informal example consider the following protocol between two players $\mathcal{P}$ and $\mathcal{V}$:

1. $\mathcal{P}$ wants to commit to a bit b. To do so, he writes b on a piece of paper and puts the paper in a box. He locks the box using a padlock.

2. $\mathcal{P}$ gives the box to $\mathcal{V}$.

3. If $\mathcal{P}$ decides to open the commitment he gives $\mathcal{V}$ the key to the padlock.

There are two basic properties of such a scheme which can be satisfied either unconditionally or relative to a complexity assumption: The *binding property* ensures that a cheating player $\mathcal{P}$ cannot change the value he has committed to at a later time, i. e., he cannot change what is inside the box. Hence, when the box is opened, we know what is revealed really was the choice that $\mathcal{P}$ committed to originally. The

hiding property guarantees that a cheating $\mathcal{V}$ cannot obtain the value $\mathcal{P}$ committed to until $\mathcal{P}$ decides to open his commitment since $\mathcal{V}$ cannot look into the box until he received the key to the padlock from $\mathcal{P}$. Note that it is not possible for a scheme to be simultaneously perfectly binding and perfectly hiding. Here, by perfect we mean that an unbounded receiver $\mathcal{V}$ gets zero information about the bit b, respectively an unbounded committer $\mathcal{P}$ can change his mind about b with probability zero.
Many different commitment schemes are known in the literature (e. g., [12, 63, 35, 64, 34]) which are based on different complexity assumptions and various notions of security. Some of them realise this basic functionality by basing it on physical processes like e. g. noisy channels or quantum mechanics, while others base it on distributing information between many players connected by a network. In the construction by Damgård and Nielsen the commitment scheme is a protocol between two players although commitment schemes may be implemented as a protocol between more players.
Their scheme is a universally composable commitment scheme with a constant expansion factor that can be instantiated in either perfectly hiding or perfectly binding versions. *Universal composability* as introduced in [17] guarantees that security is maintained even when an unbounded number of copies of the scheme are running concurrently in an adversarially controlled way. Thus it is a very strong notion of security. Furthermore, it implies non-malleability and security against adaptive adversaries. Damgård and Nielsen based the construction on a new primitive which they call *mixed commitment scheme* and which will be defined in Subsection 4.8.1. Basing the implementation on the ECPS we can obtain a mixed commitment scheme with exactly the same properties.

To intuitively explain the main ideas of [33] we think of the simplest type of commitment schemes where both committing and opening are non-interactive. To commit to a message m the committer runs an algorithm commit_K where K denotes a public key and which has as input the message m together with a uniformly random string r. Hence, he computes $c = \mathrm{commit}_K(m, r)$ and sends c to the receiver. To open the commitment, the committer sends m and r to the receiver who verifies that $c = \mathrm{commit}_K(m, r)$. Here hiding means that given just c the receiver does not learn m and binding means that the committer cannot change his mind by computing m', r' where $c = \mathrm{commit}_K(m', r')$ and $m' \neq m$.

In a *trapdoor commitment scheme* there is a piece of trapdoor information t_K which is associated to each public key K. This, if known, allows the committer to change his mind. Obviously from the existence of such trapdoor information it follows that in such a scheme the binding property cannot be satisfied unconditionally. In most trapdoor schemes it is even possible to compute from t_K commitments that can be opened in any way desired. This special type of trapdoor schemes is called *equivocable.*

Furthermore, it is possible to construct commitment schemes where a different type of trapdoor information t'_K exists, so that the knowledge of t'_K guarantees that we can efficiently compute m from $\text{commit}(m, r)$. This property immediately implies that such schemes cannot be unconditionally hiding. This type of commitment scheme is called *extractable*.

In the scheme of Canetti and Fischlin in [17] the hiding property and binding property are both fulfilled only computationally. This is owing to their construction in which it seems that a scheme is needed that is simultaneously extractable and equivocable to be universally composable. As already mentioned in [33] a new technique for the construction of universally composable commitments based on a so-called mixed commitment scheme is proposed. Basically a mixed commitment scheme is a commitment scheme which on some of the keys is perfectly hiding and equivocable, these keys are called E-keys, and on some of the keys perfectly binding and extractable, these keys are called X-keys. Note that obviously, a key cannot be both X- and E-key. The basic construction of this technique by Damgård and Nielsen is neither perfectly binding nor perfectly hiding. However, it is possible to modify their basic scheme to obtain a commitment scheme that can be instantiated in either perfectly binding or perfectly hiding.

4.8.1 Mixed Commitments

The formal definition of mixed commitment schemes given in [33] is as follows:

Definition 4.8.1. By a *mixed commitment scheme* we mean a commitment scheme commit_K with some global system key N, which determines the message space $\mathcal{M}_N$ and the key space $\mathcal{K}_N$ of the commitments. The key space contains two sets, the E-keys and the X-keys, for which the following holds:

Key generation:
One can efficiently generate a system key N along with the so-called X-trapdoor t_N. One can, given the system key N, efficiently generate random commitment keys and random X-keys. Given the system key, we can efficiently generate an E-key K along with the so-called E-trapdoor t_K.

Key indistinguishability:
Random E-keys and random X-keys are both computationally indistinguishable from random keys as long as the X-trapdoor is not known.

Equivocability:
Given an E-key K and an E-trapdoor t_K we can generate fake commitments c, distributed exactly as real commitments, which can later be opened arbitrarily, i. e., given a message m we can compute uniformly random r for which $c = \text{commit}_K(m, r)$.

Extraction:
Given a commitment $c = \text{commit}_K(m, r)$, where K is an X-key, we can, given the X-trapdoor t_N, efficiently compute m, where m is uniquely determined by the perfect binding.

Note that from key indistinguishability it follows that as long as the X-trapdoor is not known the scheme is computationally hiding for all keys. Furthermore, it implies that as long as neither the X-trapdoor nor the E-trapdoor is known the scheme is computationally binding for all keys.
For the construction based on the idea of Damgård and Nielsen a few special requirements on the mixed commitment scheme are needed:
A *special mixed commitment scheme* is a mixed commitment scheme with the following further properties: The message space $\mathcal{M}_N$ and the key space $\mathcal{K}_N$ are finite groups in which it is possible to compute efficiently. Furthermore, the ratio of E-keys over the total number of keys is negligible and the ratio of X-keys over the total number of keys is negligible close to 1. This implies that there is only a negligible fraction of keys which is neither E-key nor X-key.
As last requirement the scheme should be of a particular form. This is ensured by a transformation. The message space of the so-obtained transformed scheme is the same, but the keys are now of the form (K_1, K_2). The corresponding E-keys are

pairs of E-keys and the corresponding X-keys are pairs of X-keys. As before this construction leaves only a negligible fraction of keys which is neither E-key nor X-key. To commit to a given message m we compute $(\text{commit}_{K_1}(\bar{m}_1),\text{commit}_{K_2}(\bar{m}_2))$ where $\bar{m}_1$ and $\bar{m}_2$ are uniformly random values for which $m = \bar{m}_1 + \bar{m}_2$. If both keys are X-keys, then $\bar{m}_1$ and $\bar{m}_2$ and thus m can be computed by extraction. Note that all other properties of a special mixed commitment scheme are also fulfilled under this transformation.

4.8.2 Special Mixed Commitment Schemes Based on q One-Way Homomorphisms

In [33] the notion of *extractable q one-way homomorphisms* was introduced. It extends the definition of *q one-way homomorphism generators* from Cramer and Damgård [22]. Let G and H be finite abelian groups and let $H/f(G)$ be a cyclic group with only large prime factors in its order. The idea is to consider an easily computable homomorphism $f : G \to H$. Furthermore, without a trapdoor random elements of $f(G)$ should be computationally indistinguishable from elements chosen randomly from all over H. Note that this implies that f is hard to invert if no trapdoor is known. Given such a trapdoor associated with f it should be easy to decide about the status of an element in H. More formally, extractable q one-way homomorphisms are defined as follows [33]:
A family of extractable q one-way homomorphisms is given by a probabilistic polynomial time generator $\mathcal{G}$ which on input 1^σ where σ denotes the security parameter, outputs a description of an 8-tuple $(G, H, f, g, q, b, b', t)$, where G and H are groups, $f : G \to H$ is an efficiently computable homomorphism, $g \in H \backslash f(G), q, b, b' \in \mathbb{N}$, and t is a string called the trapdoor. Let $F := f(G)$. It is required that gF generates the factor group H/F. Furthermore, $\text{ord}(g) = |H/F|$ should be superpolynomial in the security parameter σ, e. g. 2^σ, $\text{ord}(g)|q$ and b is a public lower bound on $\text{ord}(g)$, i. e., $2 \leq b \leq \text{ord}(g) \leq q$. A generator is said to have *public order* if $b = \text{ord}(g) = q$. Also b' is superpolynomial in σ (e. g. $2^{\sigma/2}$) and its order is a public lower bound on the prime factors in $\text{ord}(g)$, i. e., all prime factors in $\text{ord}(g)$ are at least b'. In contrast to [33] we write operations in G and H additively. Finally, it is required that in both groups G and H we can add, multiply by an integer, take inverses and

sample random elements in probabilistic polynomial time given (G, H, f, g, q, b, b'). The final central requirements are as follows:

Indistinguishability: Random elements from F are *computationally indistinguishable* from random elements from H given (G, H, f, g, q, b, b').

Extractability: A generator is called *fully extractable* if given $(G, H, f, g, q, b, b', t)$ and $y = ig + f(r)$ we can compute $i \bmod \operatorname{ord}(g)$ in probabilistic polynomial time. Note that given the 8-tuple $(G, H, f, g, q, b, b', t)$ it is easy to compute $\operatorname{ord}(g)$.

q-invertibility: Given (G, H, f, g, q, b, b') and $y \in H$, it is easy to compute x so that $qy = f(x)$. Note that this does not contradict indistinguishability: since q is a multiple of $\operatorname{ord}(g)$, it is always the case that $qy \in F$.

Using such an extractable q one-way homomorphism Damgård and Nielsen [33] described a way to transform it into a special mixed commitment scheme: The message space of the commitment scheme will be $\mathbb{Z}/_{b\mathbb{Z}}$ and the key space $\mathcal{K}$ will be the group H. Hence both are finite groups in which we can compute efficiently. To commit to a message $m \in \mathbb{Z}/_{b\mathbb{Z}}$ given a key $K \in \mathcal{K}$ choose $r \in H$ randomly and compute the commitment as $c = commit_K(m, r) := mK + f(r)$. The E-keys will be the set $F = f(G)$ with corresponding E-trapdoor $t_K = f^{-1}(K)$. Due to the requirement that $\operatorname{ord}(g)$ is superpolynomial in the security parameter σ we fulfill the requirement that the number of E-keys is negligible over the total number of keys. The X-keys will be elements of the form $K = ig + f(r_K)$ where i is invertible in $\mathbb{Z}/_{\operatorname{ord}(g)\mathbb{Z}}$ with X-trapdoor t. By the requirement that all prime factors of $\operatorname{ord}(g)$ are large, the ratio of X-keys over the total number of keys is negligible close to 1 as required for a special mixed commitment scheme. Furthermore, the X-keys can be sampled efficiently given t since $\operatorname{ord}(g)$ is know.

Key indistinguishability follows directly from the requirement that random elements in $f(G)$ and random elements chosen from all of H are computationally indistinguishable. To show equivocability we assume that the E-key $K = f(r_K)$ and E-trapdoor $t_K = r_K$ is given. A fake commitment is generated as $c = f(r_c)$ for $r_c \in H$ uniformly random. Assume that we are given $m \in \mathbb{Z}/_{b\mathbb{Z}}$. Let $r := -mr_K + r_c$. Then r is uniformly random and $c = f(r_c) = f(mr_K + r) = mK + f(r) = \operatorname{commit}_K(m, r)$. This proves that $t_K = r_K$ gives equivocability, i. e., fake commitments that are distributed exactly as real commitments and can later be opened arbitrarily can be

generated. To show extractability we assume that K is an X-key, i. e., $K = ig + f(r_K)$ with i invertible in $\mathbb{Z}/_{\mathrm{ord}(g)\mathbb{Z}}$, and a commitment $c = mK + f(r) = img + f(mr_K + r)$ for $m \in \mathbb{Z}/_{b\mathbb{Z}}$ is given. Using fully extractability, i. e., using the X-trapdoor t we can compute $im \bmod \mathrm{ord}(g)$ from c and furthermore, $i \bmod \mathrm{ord}(g)$ can be computed from K. Since i is invertible we can then compute $m \bmod \mathrm{ord}(g) = m$. The transformed scheme has keys (K_1, K_2) which are pairs of keys and we commit as $\mathrm{commit}_{K_1,K_2}(m, (r_1, r_2, m_1)) = (m_1K_1 + f(r_1), m_2K_2 + f(r_2))$ where r_1, r_2 are random elements in H and $m_2 = m - m_1 \bmod q$. Hence all requirements for a special mixed commitment scheme are fulfilled.
Furthermore, Damgård and Nielsen develop efficient protocols for proving in zero-knowledge relations among committed values for the special mixed commitment scheme. See [33] for details of these protocols and for further properties and the proof of security of the commitment scheme.

4.8.3 The ECPS as an Example for a q One-Way Homomorphism

We can now show that a slightly modified version of the ECPS (see Section 4.2) is an example of a q one-way homomorphism. Hence it can be used to construct a special mixed commitment scheme as described above.

To develop the construction in an analogous manner as in [33] we have to modify the random contribution of the ECPS to build a q one-way homomorphism and hence to construct a special mixed commitment scheme: Let N be an RSA modulus (and let $s = 1$ if we consider the generalised ECPS). Furthermore, let P_1 and M be the appropriate values computed in the key generation phase of the ECPS for an elliptic curve with (a, b) as parameters. We have that $E(\mathbb{Z}/_{N^2\mathbb{Z}}) \simeq G_1 \times G_2$ where $G_1 \simeq \mathbb{Z}/_{N\mathbb{Z}}$, thus $|G_1| = N$ and G_2 is a group of order M which is hard to compute unless the factorisation of N is known, see [42]. Before defining the homomorphism f we consider the map $\Phi : (m, R) \mapsto mP_1 + NR$ where $m \in \mathbb{Z}/_{N\mathbb{Z}}$ and R is a random point on $E(\mathbb{Z}/_{N^2\mathbb{Z}})$.

Remark 4.8.2.

1. Note that we cannot directly choose a random element in $E(\mathbb{Z}/N^2\mathbb{Z})$ without the knowledge of the factors of N. To solve this we fix a number of points $Q_1, ..., Q_t$ at key generation time, i. e., when the prime factors of N are known, and then let $R = r_1Q_1 + \cdots + r_tQ_t$. By tuning t and the size of the numbers r_i for $1 \leq i \leq t$ this will generate an appropriate approximation to a uniform choice from $E(\mathbb{Z}/N^2\mathbb{Z})$. [3]

2. Note that the random contribution in this map is defined different than in the original ECPS (see Section 4.2): Instead of defining it as a fixed point Q multiplied by a random number, we take a random point R on the elliptic curve $E(\mathbb{Z}/N^2\mathbb{Z})$ multiplied by N which gives us a random element in the group G_2.

Obviously for the map Φ it holds that

$$\begin{aligned} k\Phi(m_1, R_1) + \Phi(m_2, R_2) &= (km_1 + m_2)P_1 + N(kR_1 + R_2) \\ &= \Phi(km_1 + m_2, kR_1 + R_2). \end{aligned}$$

Note that given a point $mP_1 + NR \in E(\mathbb{Z}/N^2\mathbb{Z})$ it is possible to compute $m \in \mathbb{Z}/N\mathbb{Z}$ efficiently if M (or equivalently the factorisation of N, see [42]) is known, since we can multiply by M to obtain MmP_1 and then proceed as in the decryption phase described in Section 4.2. Furthermore, if the elliptic curve Paillier assumption holds elements of the form $\Phi(0, R)$ are indistinguishable from elements of the form $\Phi(m, R)$, where R is a random point in $E(\mathbb{Z}/N^2\mathbb{Z})$ and m is any fixed element in $(\mathbb{Z}/N\mathbb{Z})^*$.

Based on this observation we can now show that relative to the elliptic curve Paillier assumption this modified version of the ECPS leads to a fully extractable generator with public order: Let N be an RSA modulus with prime factors of bit length $\sigma/2$. Let both groups G and H be $E(\mathbb{Z}/N^2\mathbb{Z})$ and let $f(R) = NR$ which is an element in $F = G_2$ and obviously efficiently computable. Let $g = P_1$, and

[3] In particular, by the Chinese Remainder Theorem it follows immediately that $t = 5$ is enough to generate random points. However, it can also be shown that $t = 4$ is enough.

let the trapdoor t be the group order M or equivalently the prime factors of N. We define $b = q = N$ to obtain a generator with public order. We have that $\mathrm{ord}(g) = N = |E(\mathbb{Z}/N^2\mathbb{Z})/G_2|$. Hence $\mathrm{ord}(g)$ as well as $b' = 2^{\sigma/2-1}$ is superpolynomial in σ. Obviously given (G, H, f, g, q, b, b') we can add points on $E(\mathbb{Z}/N^2\mathbb{Z})$ (see Section 4.1), multiply them by integers, take inverses of points in $E(\mathbb{Z}/N^2\mathbb{Z})$ in probabilistic polynomial time. Additionally, we can sample random elements in $E(\mathbb{Z}/N^2\mathbb{Z})$ as noted above. Furthermore, the elliptic curve Paillier assumption guarantees indistinguishability. If we are given (G, H, f, g, q, b, b') and additionally the trapdoor $t = M$ we can compute $i \bmod N$ given $ig + f(R) = iP_1 + NR$ using the decryption algorithm, i. e., multiplying by M. This shows that the generator is fully extractable. The last property that has to be verified is q-invertibility which follows immediately from the definition.

Hence, using the result of Damgård and Nielsen [33] as described in Subsection 4.8.2 we obtain the following theorem:

Theorem 4.8.3. *Based on the ECPS it is possible to build a special mixed commitment scheme with message space $\mathbb{Z}/N\mathbb{Z}$ with proofs of relations of the form $m = f(m_1, m_2, \ldots, m_l)$ where f is a Boolean predicate and $l = O(\log(k))$ and proofs of additive and multiplicative relations modulo N.*

Chapter 5

Conclusions and Open Questions

This thesis dealt with homomorphic cryptosystems and their applications. In Chapter 2 we have efficiently constructed an algebraically homomorphic cryptosystem given a homomorphic cryptosystem on a special non-abelian group. Designing such an algebraically homomorphic cryptosystem has been an open problem for more than 20 years since a positive result leads to efficient and simple solutions to several cryptographic protocols. We were able to partially solve this long standing open problem by reducing it to the search for special efficient homomorphic cryptosystems.

The solutions presented in Chapter 3 were based on the idea of considering branching programs instead of circuits as a computational model for functions. By such means we were able to provide non-interactive and provably secure protocols to encrypt functions given by polynomial branching programs in such a way that they are still executable. Hence we enlarge the class of encryptable functions from NC^1 to polynomial branching programs. This is an improvement of existing solutions and - to the best of our knowledge - there are no concrete functions known that cannot be represented by polynomial branching programs. Future research will show if there exist solutions based on homomorphic schemes that are independent of the size of the underlying function. Finally, owing to this enlargement the question arises whether algebraically homomorphic cryptosystems are still necessary for specific applications or whether homomorphic cryptosystems are sufficient.

In Chapter 4 we constructed a threshold decryption version of the elliptic curve Paillier scheme. This version is especially suited for many applications since various powerful auxiliary protocols can be built based on it. Our protocols are the first elliptic curve versions of this kind. It is interesting if there exist further elliptic curve versions of the Paillier scheme that are as powerful as our scheme, i. e. that lead to the same protocols. Furthermore, one could analyse our results to determine the specific properties required of the underlying cryptosystem to obtain all mentioned auxiliary protocols.

Bibliography

[1] M. Abadi, J. Feigenbaum, *Secure Circuit Evaluation - a Protocol based on Hiding Information from an Oracle*, Journal of Cryptology, Vol. 1, No. 2, 1990, pp. 1–12

[2] A. Adelsbach, S. Katzenbeisser, A. Sadeghi, *Cryptography Meets Watermarking: Detecting Watermarks with Minimal or Zero Knowledge Disclosure*, In Proceedings of the European Signal Processing Conference 2002, Toulouse, France, 2002

[3] F. Bao, *Cryptanalysis of a Provable Secure Additive and Multiplicative Privacy Homomorphism*, Workshop on Coding and Cryptography, 2003, Versailles

[4] D. A. Barrington, *Bounded-with Polynomial-size Branching Programs Recognize Exactly those Languages in NC^1*, In Proceedings of the 18th ACM Symposium on the Theory of Computing, 1986, pp. 1–5

[5] D. M. Barrington, H. Straubing, D. Therien, *Non-uniform Automata over Groups*, Information and Computation, Vol. 89, No. 2, 1990, pp. 109–132

[6] M. Bellare, P. Rogaway, *Random Oracles are Practical: a Paradigm for Designing Efficient Protocols*, First ACM Conference on Computer and Communications Security, 1993, pp. 62–73

[7] J. C. Benaloh, *Verifiable Secret-Ballot Elections*, Ph. D. Thesis, Technical Report 561, Yale University, Department of Computer Science, 1988

[8] J. Benaloh, *Dense Probabilistic Encryption*, In Proceedings of the Workshop on Selected Areas of Cryptography, 1994, pp. 120–128

[9] J. Benaloh, M. Yung, *Distributing the Power of a Government to Enhance the Privacy of Voters*, In Proceedings of the 5th ACM Symposium on Principles of Distributed Computing (PODC'86), New York, 1986, pp. 52–62

[10] M. Ben-Or, R. Cleve, *Computing Algebraic Formulas Using a Constant Number of Registers*, SIAM Journal on Computing, Vol. 21, No. 1, 1992, pp. 54–58

[11] I. Blake, G. Seroussi, N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, Lecture Note Series 265, Cambridge University Press, 1999

[12] M. Blum, *Coin Flipping by Telephone*, IEEE Spring COMPCOM, 1982, pp. 133–137

[13] Boneh, Lipton, *Searching for Elements in Black Box Fields and Applications*, In Proceedings of Crypto '96, LNCS 1109, Springer-Verlag, 1996, pp. 283–297

[14] G. Brassard, D. Chaum, C, Crépeau, *Minimum Disclosure Proofs of Knowledge*, Journal of Computer and System Sciences, Vol. 37, No. 2, pp. 156-189, 1988

[15] E. F. Brickell, Y. Yacobi, *On Privacy Homomorphisms*, Advances in Cryptology - EUROCRYPT 1987, LNCS 304, Springer-Verlag, 1987, pp. 117–126

[16] C. Cachin, J. Camenisch, J. Kilian, J. Müller, *One-round Secure Computation and Secure Autonomous Mobile Agents*, In Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP), LNCS 1853, Springer-Verlag, 2000, pp. 512–523

[17] R. Canetti, M. Fischlin, *Universally Composable Commitments*, Advances in Cryptology - CRYPTO '01, LNCS 2139, Springer-Verlag, 2001, pp. 19–40

[18] D. Catalano, R. Gennaro, N. Howgrave-Graham, *The Bit Security and Paillier's Encryption Scheme and its Applications*, Advances in Cryptology - EUROCRYPT 2001, LNCS 2045, Springer-Verlag, 2001, pp. 229–243

[19] D. Catalano, R. Gennaro, N. Howgrave-Graham, *Paillier's Trapdoor Function Hides up to $O(n)$ Bits*, Journal of Cryptology, Vol. 15, 2002, pp. 251–269

[20] D. Catalano, R. Gennaro, N. Howgrave-Graham, P. Q. Nguyen, *Paillier's Cryptosystem Revisited*, In Proceedings of the 8th ACM Conference on Computer and Communication Security, 2001, pp. 206–214

[21] J. Cohen, M. Fischer, *A Robust and Verifiable Cryptographically Secure Election Scheme*, In Proceedings of the 26th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 1985, pp. 372–382

[22] R. Cramer, I. Damgård, *Zero-Knowledge for Finite Field Arithmetic. Or: Can Zero-Knowledge be for Free?*, In Proceedings of CRYPTO '98, LNCS 1462, Springer-Verlag, 1998, pp. 424–441

[23] R. Cramer, I. Damgård, U. Maurer, *General Secure Multiparty Computation from any Linear Secret-sharing Scheme*, In Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, Springer-Verlag, 2000, pp. 316–334

[24] R. Cramer, I. Damgård, J. Nielsen, *Multiparty Computation from Threshold Homomorphic Encryption*, In Proceedings of EUROCRYPT '01, LNCS 2045, pp. 280-299, 2001

[25] R. Cramer, I. Damgård, B. Schoenmakers, *Proofs of Partial Knowledge*, In Proceedings of CRYPTO '94, LNCS 839, Springer-Verlag, 1994, pp.174–187

[26] R. Cramer, S. Fehr, Y. Ishai, E. Kushilevitz, *Efficient Multi-Party Computation over Rings*, In Proceedings of EUROCRYPT 2003, LNCS 2656, Springer-Verlag, 2003

[27] R. Cramer, R. Gennaro, B. Schoenmakers, *A Secure and Optimally Efficient Multi-Authority Election Scheme*, In European Transactions on Telecommunications, Vol. 8, No. 5, 1997, pp. 481-490

[28] I. Damgård, *On the Existence of Bit Commitment Schemes and Zero-knowledge Proofs*, Advances in Cryptology - CRYPTO '89, LNCS 435, Springer-Verlag, 1989, pp. 17–29

[29] I. Damgård, *On Σ-Protocols*, Cryptologic Protocol Theory Course, 2002, URL: `http://www.daimi.au.dk/~ivan/CPT.html`

[30] I. Damgård, M. Jurik, *A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System*, Public Key Cryptography (PKC 2001), LNCS 1992, Springer-Verlag, 2001, pp. 119-136

[31] I. Damgård, M. Jurik, *A Length-Flexible Threshold Cryptosystem with Applications*, In Proceedings of the 8th Australasian Conference on Information Security and Privacy (ACISP 2003), LNCS 2727, Springer-Verlag, 2003

[32] I. Damgård, M. Jurik, J. Nielsen, *A Generalisation of Paillier's Public-Key System with Applications to Electronic Voting*, Special issues on Financial Cryptography, International Journal on Information Security (IJIS), Springer-Verlag, 2003

[33] I. Damgård, J. Nielsen, *Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor*, Advances in Cryptology - CRYPTO 2002, LNCS 2442, Springer-Verlag, 2002, pp. 581–596

[34] G. Di Crescenzo, J. Katz, R. Ostrovsky, A. Smith *Efficient and Perfectly-Hiding Non-Interactive, Non-malleable Commitment*, Advances in Cryptology - EUROCRYPT 2001, LNCS 2045, 2001, pp. 40–59

[35] D. Dolev, C. Dwork, M. Naor, *Non-malleable Cryptography*, SIAM Journal on Computing, Vol. 30, No.2, 2000, pp. 391-437

[36] J. Domingo-Ferrer, *A Provably Secure Additive and Multiplicative Privacy Homomorphism*, In Proceedings of the 5th International Conference on Information Security (ISC 2002), LNCS 2433, Springer-Verlag, 2002, pp. 471–483

[37] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms*, Advances in Cryptology - Proceedings of CRYPTO '84, LNCS 196, 1985, pp. 10–18

[38] J. Feigenbaum, M. Merritt, *Open Questions, Talk Abstracts, and Summary of Discussions*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 2, 1991, pp. 1–45

[39] M. Fellows, N. Koblitz, *Combinatorial Cryptosystems Galore!*, In Finite Fields: theory, applications and algorithms, Contemporary Mathematics, Vol. 168, Las Vegas, 1993, pp. 51–61

[40] A. Fiat, A. Shamir, *How To Prove Yourself: Practical Solutions to Identification and Signature Problems*, Advances in Cryptology - CRYPTO '86, LNCS 263, Springer-Verlag, 1987, pp. 186–194

[41] P. Fouque, G. Poupard, J. Stern, *Sharing Decryption in the Context of Voting or Lotteris*, Financial Cryptography 2000, LNCS 1962, Springer-Verlag, 2000

[42] S. D. Galbraith, *Elliptic Curve Paillier Schemes*, Journal of Cryptology, Vol. 15, No. 2, 2002

[43] Z. Galil, S. Haber, M. Yung, *Cryptographic Computation: Secure Fault-tolerant Protocols and the Public-key Model*, Advances in Cryptology - CRYPTO '87, LNCS 293, Springer-Verlag, 1988, pp. 135-155

[44] S. Goldwasser, S. Micali, *Probabilistic Encryption* & *How to Play Mental Poker Keeping Secret All Partial Information*, Proceedings of the 14th ACM Symposium on the Theory of Computing, 1982, pp. 270-299

[45] S. Goldwasser, S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences, Vol. 28, No. 2, 1984, pp. 270–299

[46] O. Goldreich, S. Micali, A. Wigderson, *How to Play any Mental Game*, ACM Symposium on Theory of Computing (STOC), 1987, pp. 218–229

[47] O. Goldreich, S. Micali, A. Wigderson, *Proofs that Yield Nothing but their Validity or All Languages in NP have Zero-Knowledge Proof Systems*, Journal of the ACM, Vol. 38, No. 1, 1991, pp. 691-729

[48] P. Golle, M. Jakobsson, A. Juels, P. Syverson, *Universal Re-encryption for Mixnets*, In Proceedings of the RSA Conference Cryptographer's Track '04, LNCS 2964, Springer-Verlag, 2004

[49] D. Grigoriev, I. Ponomarenko, *Homomorphic Public-key Cryptosystems and Encrypting Boolean Circuits*, Workshop on Codes and Cryptography, INRIA, Versailles, 2003

[50] D. Grigoriev, I. Ponomarenko, *Homomorphic Public-key Cryptosystems over Groups and Rings*, http://arxiv.org/abs/cs.CR/0309010

[51] J. Groth, *Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems*, In Proceedings of the First Theory of Cryptography (TCC 2004), LNCS 2951, Springer-Verlag, 2004, pp. 152–170

[52] J. W. P. Hirschfeld, *Projective Geometries over Finite Fields*, Oxford Mathematical Monographs, Clarendon Press, Oxford, p. 30, 1979

[53] Y. Ishai, E. Kushilevitz, *Private Simultaneous Messages Protocols with Applications*, In Proceedings of the Fifth Israel Symposium on Theory of Computing and Systems (ISTCS 1997), IEEE Computer Society, 1997, pp. 174–183

[54] Y. Ishai, E. Kushilevitz, *Randomizing Polynomials: A new Representation with Applications to Round-efficient Secure Computation*, In Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000), 2000, pp. 294–304

[55] Y. Ishai, E. Kushilevitz, *Perfect Constant-Round Secure Computation via Perfect Randomizing Polynomials*, In Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP 2002), LNCS 2382, Springer-Verlag, 2002, pp. 244–256

[56] S. Jarecki, A. Lysyanskaya, *Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures*, Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, Springer-Verlag, 2000, pp. 221–242

[57] M. Jurik, *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*, Ph.D. Thesis, Department of Computer Science, University of Aarhus, Denmark, 2003

[58] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. S. Kang, C. Park, *New Public-key Cryptosystem Using Braid Groups*, Advances in Cryptology - CRYPTO 2000, LNCS 1880, Springer-Verlag, 2000, pp. 166–183

[59] N. Koblitz, *Algebraic Aspects of Cryptography*, Algorithms and Computation in Mathematics, Vol. 3, Springer-Verlag, New York, 1998.

[60] Helger Lipmaa, *Verifiable Homomorphic Oblivious Transfer and Private Equality Test*, Advances in Cryptology - Asiacrypt 2003, LNCS 2894, Springer-Verlag, 2003

[61] L. V. Ly, *Polly Two - A Public-Key Cryptosystem based on Polly Cracker*, Ph.D. Thesis, Ruhr-Universität Bochum, 2002

[62] D. Naccache, J. Stern. *A New Public Key Cryptosystem Based on Higher Residues*, In Proceedings of the 5th ACM Conference on Computer and Communications Security, 1998, pp. 59–66

[63] M. Naor, *Bit Commitment Using Pseudo-Randomness*, Journal of Cryptology, Vol. 4, 1991, pp. 151–158

[64] M. Naor, R. Ostrovsky, R. Venkatesan, M. Yung, *Perfect Zero-Knowledge Arguments for NP Can be Based on General Complexity Assumptions*, Advances in Cryptology - CRYPTO'92, LNCS 740, 1992, pp. 196–214

[65] T. Okamoto, S. Uchiyama, *A New Public-Key Cryptosystem as Secure as Factoring*, Advances in Cryptology - EUROCRYPT 1998, LNCS 1403, Springer-Verlag, 1998, pp. 308-318

[66] S.-H. Paeng, K.-C. Ha, J. H. Kim, S. Chee, C. Park, *New Public Key Cryptosystem Using Finite Non Abelian Groups*, Advances in Cryptology - CRYPTO 2001, LNCS 2139, Springer-Verlag, 2001, pp. 470–485

[67] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology - EUROCRYPT 1999, LNCS 1592, Springer-Verlag, 1999, pp. 223–238

[68] P. Paillier, *Trapdooring Discrete Logarithms on Elliptic Curves over Rings*, Advances in Cryptology - ASIACRYPT 2000, LNCS 1976, Springer-Verlag, 2000, pp. 573–584

[69] P. Paillier, D. Pointcheval, *Efficient Public-Key Cryptosystems Provably Secure Against Active Adversaries*, Advances in Cryptology - ASIACRYPT '99, LNCS 1716, Springer-Verlag, 1999, pp. 165–179

[70] B. Pfitzmann, M. Waidner, *Anonymous Fingerprinting*, Advances in Cryptology - EUROCRYPT 1997, LNCS 1233, Springer-Verlag, 1997, pp. 88–102

[71] G. Poupard, J. Stern, *Fair Encryption of RSA Keys*, Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, Springer-Verlag, 2000

[72] D. K. Rappe, *Algebraisch homomorphe Kryptosysteme*, Diploma Thesis, University of Dortmund, Germany, 2000
URL: `http://www.matha.mathematik.uni-dortmund.de/~rappe/`

[73] R. Rivest, L. Adleman, M. Dertouzos, *On Data Banks and Privacy Homomorphisms*, Foundations of Secure Computation, Academic Press, 1978, pp. 169–177

[74] R. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol. 21, No. 2, 1978, pp. 120–126

[75] T. Sander, C. F. Tschudin, *Towards Mobile Cryptography*, IEEE Symposium on Security & Privacy '98, Oakland, California, 1998, pp. 215–224

[76] T. Sander, C. F. Tschudin, *Protecting Mobile Agents Against Malicious Hosts*, in Mobile Agents and Security, LNCS 1419, 1998

[77] T. Sander, A. Young, M. Yung, *Non-Interactive CryptoComputing for* NC^1, In Proceedings of the 40th Annual Symposium on Foundations of Computer Science, IEEE, 1999, pp. 554–567

[78] A. Shamir, *How to Share a Secret*, Communications of the ACM, vol. 22, no. 11, 1979, pp. 612–613

[79] V. Shoup, *Practical Threshold Signatures*, Advances in Cryptology - EUROCRYPT 2000, LNCS 1807, Springer-Verlag, 2000, pp. 207–220

[80] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics Vol. 106, Springer-Verlag, 1986

[81] D. Wagner, *Cryptanalysis of an Algebraic Privacy Homomorphism*, In Proceedings of the 6th International Conference on Information Security (ISC 2003), LNCS 2851, Springer-Verlag, 2003,
URL: `http://www.cs.berkeley.edu/~daw/papers/` (revisted version)

[82] N. R. Wagner, M. R. Magyarik, *A Public Key Cryptosystem Based on the Word Problem*, Advances in Cryptology - CRYPTO '84, LNCS 196, Springer-Verlag, 1985, pp. 19–36

[83] I. Wegener, *The Complexity of Boolean Functions*, Wiley-Teubner Series in Computer Science, 1987

[84] A. Yao, *Protocols for Secure Computation* (extended abstract), In 23rd Annual Symposium on Foundations of Computer Science, IEEE, 1982, pp. 160–164

[85] A. Yao, *How to Generate and Exchange Secrets*, In 27th Annual Symposium on Foundations of Computer Science (FOCS 1986), IEEE, 1986, pp. 162-167

Index