Electronic Remote Voting

Diplomarbeit

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Eingereicht von: Daniel Schliebner,
geb. am 08.06.1986 in Berlin.

Eingereicht am: 29.11.2011

Betreuer: Prof. Dr. Johannes Köbler
Prof. Dr. Ernst-Günter Giessmann

Contents

1. Introduction

Today, the internet is accessible from nearly everywhere yielding a trend in which a lot of activities are carried out online such as electronic shopping, electronic tax return and internet banking among others. This omnipresence of the internet however is due to the fact that smartphones and other mobile devices having network access become affordable and an explanation is the fact that in modern society the saving of time is increasingly important for each individual. It is thus reasonable to assume that electronic elections which do not require the presence of a trusted and monitored physical place such as a voting booth but the possibility to vote from everywhere only using an internet connection could make voting more convenient such that this considerably encourages voter turnouts in modern elections (this paradigm is called *remote voting*). This assumption can already be monitored in today's paper based elections in Germany, where each voter is able to request an absentee ballot. Strictly speaking an absentee ballot does not fulfill the principles of a democratic election since anyone can sell its vote or even can be coerced. But however a large amount of absentee ballot applications is approved with the aim to increase voter turnouts. Another advantage of electronic elections is that it is reasonable to be easier to perform large scale elections and to produce exact results immediately without the need of extrapolations.

The main intention of this thesis is to give a brief overview of the electronic voting topic and to present approaches on how to securely realize electronic remote voting. Concretely this work is structured as follows.

The first chapter is intended to give a brief historical overview of electronic voting. Thereby, in Section 2.1 we discuss advantages of electronic elections, how this paradigm came up and which security properties are necessary to be fulfilled. Section 2.2 is then devoted to the cryptographic point of view. Considering a single voter V there exist two intrinsic approaches how to cast a vote v, namely V can send either v anonymously to a tallier or he can send v encrypted. For the last approach it is necessary that the talliers cannot decrypt single votes since this would affect the anonymity of V. We present the three main techniques used in recent scientific literature which pick up these approaches to ensure an electronic remote election. These techniques include blind-signatures, mix-nets and homomorphic encryption and beside the formal presentation of these paradigms Section 2.2 additionally discusses advantages and disadvantages of them, while it will be the main purpose of the forthcoming sections to analyze homomorphic encryption based voting schemes.

As time proceeded, a lot of electronic voting schemes were published and only few of them provided security proofs. It is thus a reasonable approach to evaluate security of voting

protocols in a framework which enables us to compare easily. Groth [Gro04] therefore proposed to consider the Universal Composability (UC) framework by Canetti [Can01]. Of course, this framework enables us to use secure protocols concurrently within larger protocols due to the fact that it comes with a universal composition theorem (Theorem 1). The security definition used within this framework is simulation based and compares the execution of a protocol in a "real world" with the execution of an ideal functionality in an "ideal world". Consequently, the second chapter is devoted to the discussion of evaluating voting schemes in the UC framework. Hence, after introducing the UC framework in Section 3.1 the motivation of the forthcoming sections is to construct a UC-secure voting protocol that securely realizes an ideal voting functionality $\mathcal{F}_{\text{Voting}}$ (Definition 20). Consequently, Section 3.2 presents preliminary theoretical backgrounds (such as threshold encryption and Σ-protocols) used to construct a voting protocol (Definition 22) based on homomorphic threshold encryption within Section 3.3. Finally, we prove that this protocol indeed UC-realizes the ideal voting functionality $\mathcal{F}_{\text{Voting}}$ (Theorem 4). We point out that this proof was originally presented by Groth in [Gro04]. Unfortunately, the voting protocol in [Gro04] communicates with an ideal message board functionality $\mathcal{F}_{\text{KM}}$ which additionally provides a key generation without giving information if this functionality can be UC-realized. Indeed it turns out that the UC-realizability depends on the underlying cryptosystem due to the fact that the distributed key generation schemes does. Namely, we prove in Section 3.4 that we can extend the voting protocol of [Gro04] to a voting protocol that UC-realizes $\mathcal{F}_{\text{Voting}}$ with only having access to some common reference string for the class of discrete logarithm based cryptosystems that use a value $y = g^x$ as public key. However, this construction cannot be performed for other public key cryptosystems, whence we additionally discuss the difficulties for cryptosystems that use a RSA-modulus $N = pq$ as public key and hence factorization as trapdoor. Moreover we present approaches on how to distributedly compute public keys for Pallier encryption which is well-suited for electronic voting based on homomorphic encryption.

The last chapter is then devoted to incoercible protocols and applications to electronic voting. The term of incoercible protocols was first noted by Benaloh in [BT94] and it can be motivated as follows. During an electronic election, a voter may obtain some kind of receipt during the voting phase which enables him to prove that he has cast a specific vote. This receipt however enables an adversary to put threat on a voter. Namely, the adversary may coerce the voter to cast a specific vote and due to the fact that the voter has an receipt, the adversary can verify if the voter was obeying or deceiving. To overcome this problem, Unruh and Müller-Quade [UMQ10] proposed a framework which is an extension of the UC framework by Canetti. Namely, the UC/c framework provides a security definition that implies the incoercibility of a voting protocol. Hence, Section 4.1 gives a brief introduction to this framework while we describe in Section 4.2 how to apply this to the setting of electronic elections. Unfortunately it turns out that the UC/c-realizability is difficult to obtain and a possible approach is the concept of deniable cryptography by Canetti et al. [CDNO96] which we introduce in Section 4.3.

The study of legal aspects of electronic voting is an autarkic scientific field of research and theoretical cryptographic approaches cannot dissociate from this important topic since their practical benefit relies heavily on legal aspects within the country in question. We stress that it is still very difficult to bring legally binding electronic elections in line with local law, e. g. especially in Germany electronic elections seem to infringe the democratic basic rights and without political engagement it will be difficult to manifest e-voting not only in Germany but in many other countries. However, although these aspects are meaningful to discuss and study, we are concentrated on the theoretical and cryptographic background within this thesis but want to point out that one has to be aware of the juristic difficulties coming along with electronic elections.

Einleitung

In der heutigen Zeit ist das Internet nahezu von überall aus erreichbar. Infolge dessen zeichnet sich in der Gesellschaft ein Trend ab, nach dem zunehmend mehr Aufgaben von unterwegs oder von zu Hause aus bewältigt werden. Alltägliche Beispiele sind unter anderem Internet Banking, die elektronische Steuererklärung oder Online Shopping. Ein Grund für die stetige Verbreitung und Allgegenwärtigkeit des Internets liegt sicherlich darin, dass in jüngerer Vergangenheit Smartphones und andere mobile, internetfähige Geräte erschwinglich geworden sind. Insbesondere ist es in der derzeitigen Gesellschaft immer wichtiger, möglichst zeiteffektiv zu arbeiten, um den hohen Ansprüchen der modernen Arbeitswelt gerecht zu werden. Nicht zuletzt auch deshalb ist es nachvollziehbar anzunehmen, dass elektronische Wahlen, welche keinen physisch abgegrenzten und überwachten Ort benötigen (wie z. B. eine Wahlkabine), im Gegenzug aber das Wählen von jedem Ort mit Internetzugang gewährleisten, wesentlich die Wahlbeteiligung steigern können. Ein solches Szenario, indem also kein physisch abgegrenzter und sicherer Ort existiert, an dem der Wähler seine Stimme abgibt, bezeichnen wir im Folgenden als *Remote Voting*. Die Annahme, dass ein solcher Typ von Wahl die Wahlbeteiligung steigern kann, lässt sich bereits an der derzeit in Deutschland praktizierten Briefwahl beobachten. Streng genommen erfüllt bereits die Briefwahl nicht mehr die Prinzipien einer demokratischen Wahl, weil hierdurch bereits die Möglichkeit gegeben ist, dass Wahlstimmen gekauft oder erpresst werden können. Dennoch werden immer häufiger Briefwahlanträge genehmigt und dies vermutlich aus der Motivation heraus, die Wahlbeteiligung zu erhöhen. Nicht zuletzt bieten elektronisch durchgeführte Wahlen den offensichtlichen Vorteil, dass abgegebene Stimmen wesentlich schneller und mit geringerem Aufwand ausgezählt werden können, als das in Papierwahlen der Fall ist. Insbesondere eliminiert das elektronische Auszählen die Notwendigkeit von Hochrechnungen.

Der Anspruch dieser Arbeit ist daher, einen kurzen Überblick über das Thema elektronischer Wahlen, vor allem aus theoretischer und kryptographischer Sicht, zu geben und darüber hinaus Ansätze zu präsentieren, wie elektronische Wahlen nach dem Prinzip des *Remote Voting* beweisbar sicher durchgeführt werden können.

Zuletzt möchten wir anmerken, dass das Studium juristischer Aspekte elektronischer Wahlen ein eigenständiges wissenschaftliches Forschungsgebiet darstellt, sich theoretische kryptographische Ansätze aber nicht gänzlich davon distanzieren können. Dies liegt vor allem darin begründet, dass der praktische Nutzen theoretischer Techniken nicht zuletzt vor allem davon abhängt, ob und inwieweit diese in der Praxis in einem Staat Anwendung finden können und dürfen. Wir stellen zudem fest, dass dies im allgemeinen bisher nur schwer zu erreichen ist. Generell besteht nämlich die Problematik, dass *rechtlich verbindliche* elektronische Wahlen nur schwer bis gar nicht gesetzeskonform zu realisieren

sind. Beispielsweise verletzen elektronische Wahlen in Deutschland in jedem Sinne (d. h. also insbesondere auch Wahlen mithilfe von Wahlcomputern) die demokratischen Grundrechte und ohne politisches Engagement scheint es bisher unmöglich, elektronische Wahlen nicht nur in Deutschland, sondern auch in anderen Ländern, zu manifestieren. Auch wenn es wichtig ist, die soeben erläuterten gesellschaftspolitischen Aspekte zu diskutieren, beschränken wir uns in dieser Arbeit auf den theoretischen Hintergrund solcher Wahlen, stellen aber fest, dass darüber hinaus leider auch juristisch und politisch noch viele wichtige Fragen offen sind.

2. Electronic Voting Schemes: An Overview

Within this chapter we give a brief introduction into the electronic remote voting scenario. Section 2.1 is intended to give an informal overview of different aspects on electronic voting along with a short historical introduction. In Section 2.2 we present the three common cryptographic techniques based on blind signatures, mix-nets and homomorphic encryption, which are used in modern remote voting protocols.

Let us first briefly introduce some notations. By a *cryptosystem* we mean a tuple (E, D) of an encryption function $E : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{C}$ mapping a key $k_1 \in \mathcal{K}$ and a plaintext $m \in \mathcal{M}$ on a ciphertext $c \in \mathcal{C}$ and a decryption function $D : \mathcal{K} \times \mathcal{C} \longrightarrow \mathcal{M}$ mapping a key $k_2 \in \mathcal{K}$ and a ciphertext c to the corresponding plaintext m. We stress that, if the cryptosystem is a public key system, we sometimes omit the public keys in the notation of the encryption and decryption function or write these as index.

Given a ring R (e. g. $R = \mathbb{Z}, \mathbb{Z}_N$ etc.) we write $R[z]$ for the ring consisting of polynomials $p(z) = \sum_{i=0}^{k} p_i z^i$ with coefficients $p_i \in R$ of arbitrary but finite degree.

Finally we write $r \in_R G$ if a number r is chosen from a finite set G uniformly at random and for Euler's totient function we use the symbol φ.

2.1. An Informal Overview

Electronic voting is not a new paradigm. Already within the 1980s and in the beginning of the 90s, electronic voting schemes were discussed in the literature and initial ideas were subsumed in first papers on electronic voting schemes, see [JCJ05, Section 1.1]. While first electronic voting schemes and practical implementations followed the paradigm of supervised voting (i. e. they assumed the presence of a voting booth, human supervision or similar authorities), the last years trend is different as a new lifestyle in modern society is noticeable. Namely, the growing availability of mobile internet and smartphones boost the wish to perform interactions from every place at every time.

Consequently, there naturally has arisen a new worthwhile approach for realizing secret ballots. Namely, the concept of *remote voting* became progressively more famous in both, scientific literature and practice. (We want to stress that the term "e-voting" is used as a synonym for "remote electronic voting" within this thesis.) As an example, Estonia was the first nation, which conducted legally binding electronic elections in 2005 [CCM08, p. 3]. Another example is the Rijnland Election System (RIES), used for national elections 2006 in the Netherlands [Pie07, Section 7.2]. Both examples, the Estonian and the Dutch, are based on public key cryptography together with the usage of digital signatures and, as we will see in the successive section, these systems leave several security issues open.

However, there naturally arise two questions:

1. What are the advantages of electronic remote voting?

2. What kinds of security do we have to claim from cryptographic protocols realizing electronic elections?

Of course, the answer to the first question is easy: counting or tallying votes manually is not future-proof or fail-safe and for large scale elections quite slow. Moreover, electronic elections may guarantee new properties an election could have. As an example it turns out that *verifiability* or *universal verifiability* can be realized in many electronic voting schemes, where the former means, roughly speaking, that every voter can easily check that its vote was counted correctly and the latter that he can verify the overall outcome of the election, too. These properties are difficult to guarantee in paper-based elections and might therefore be worthwhile to obtain.

The second question leads us to the main problem of electronic voting schemes, which is, how trustworthy it has to be, i. e. which security properties we have to assume and to prove when designing an electronic voting protocol. Albeit until now, no agreement which security aspects to assume is reached, it is possible to extract the most common requirements which are [Pie07, p. 39]

1. *Availability*: all eligible voters can vote (ideally with the same effort);
2. *Authenticity*: only eligible users vote;
3. *Correctness*: all counted votes are valid and all valid votes are counted;
4. *Verifiability*: results can be scrutinized by all involved voters;
5. *Secrecy*: a voter can (or must) keep his vote secret.

As we will see, this distinction has still not enough granularity but the difficulty in determining the security requirements of an electronic voting system is already visible. Namely, it seems to be difficult to realize both, verifiability and secrecy.
In addition to the five requirements above, a lot more occur in the literature, the most important of which are [Gro04, p. 2]

1. *Privacy*,
2. *Accuracy*,
3. *Fairness*,
4. *Robustness*,
5. *Universal Verifiability*,
6. *Incoercibility* and
7. *Receipt-freeness*.

Let us briefly sketch the ideas behind these requirements, where we want to discuss incoercibility and receipt-freeness separately. Privacy can be seen as a synonym for the secrecy property above, while fairness and robustness are embraced by the term availability. Moreover, accuracy is embraced by the term correctness meaning that all eligible votes are counted correctly with only a (possibly small) error tolerance, depending on the application. Universal verifiability is an extension of the verifiability requirement and

states, that every voter can verify the overall outcome of the election itself, i. e. he can calculate the outcome.

As we have mentioned above, remote voting is a new desirable paradigm but an immediate threat which comes up is the possibility of vote buying or coerced votes. In traditional elections it is not possible by an adversary to coerce or buy an eligible voter in order to let him cast a specific vote. The reason is that the voter may abstain from voting or even cast any vote without obtaining any receipt. Thus, the voter is in any case not able to reveal anything about his vote and consequently, the adversary cannot threat some reprisal on the voter. For this reason, we may think about receipt-free electronic voting schemes in which the voter does not get or even can acquire a receipt. This requirement was first introduced by Benaloh and Tuinstra [BT94]. If, moreover, the voter cannot prove how he voted, *even if he colludes with the adversary*, and the adversary does not learn anything from the vote during the voting process, we informally speak of an incoercible voting system [CCM08, p. 4].

2.2. Cryptographic Approaches to Secure e-Voting

This section is devoted to the presentation of the three basic techniques used to realize electronic voting protocols.

2.2.1. Blind Signatures

Blind signatures are a well-known concept in cryptography where a message m of a person A is signed by another person B (the *signing authority*) such that B gains no information about the message m. A simple example is the following based on RSA.

Example 1 (Blind Signatures)**.** *Let $n = pq$ be a RSA modulus. A has a message $m \in \mathbb{Z}_n$ that he wants to be signed blindly by B. Let $e \in \mathbb{Z}^*_{\varphi(n)}$ be the public key and $d \in \mathbb{Z}^*_{\varphi(n)}$ the private key of B such that $ed \equiv_{\varphi(n)} 1$. Then we proceed as follows:*

1. *A chooses $r \in_R \mathbb{Z}^*_n$ at random and sends $x := m \cdot r^e \bmod n$ to B.*
2. *B sends $y := x^d \bmod n$ back to A.*
3. *A calculates $z := y \cdot r^{-1} = m^d \bmod n$ which is the signature.*

The last step is clearly valid since $ed = 1 \bmod n$.

An e-voting scheme based on blind signatures now works as follows. First, each voter obtains a pseudonym z from the signing authority that enables him to cast a vote. This is done by using a blind signature: the voter chooses some message m of a predefined structure, e. g. some published identifier corresponding to the election, and the authority then returns a y from which the signature z of the message m is calculated; this is the pseudonym. Thereby the authority has to verify that the voter is eligible and that he does not already obtained a pseudonym. Finally, the voter will cast a vote v by sending (z, v) anonymously (i. e. through an anonymous channel) to a tallier. An example can be found within [FOO93].

Advantages

The big advantage of schemes based on blind signatures described above is their efficiency in both, the voting phase and the phase where the votes are counted to give the overall outcome of the election. However, as we will see, this efficiency becomes manifest in a series of disadvantages. This has the effect that schemes based on blind signatures in the sense above have generated less interest within recent literature.

Disadvantages

Common voting schemes based on blind signatures fail to guarantee universal verifiability since it is not possible for outside participants to verify that only eligible votes where counted and all counted (or even tallied) votes where casted by an eligible voter. This is mostly due to the fact that such schemes *fail to manage voters who abstain from voting*. In such a case a corrupted authority may impersonate such a voter and thus can cast its own vote. Moreover, authorities may provide some distinguished voters with more than one pseudonym which enables those voters to cast more than one vote. However, this all contradicts universal verifiability since no outside participant is able to notice such activities.

Another disadvantage or even problematic point is the anonymous or untraceable channel required to cast a vote which is difficult to achieve in practice and the idea to broadcast votes together with an identity in an unencrypted way seems not to be a trustworthy approach.

Conclusions

The confrontation of advantages and disadvantages shows that bare blind signature e-voting schemes as described above seem not to be very practical albeit they guarantee the best efficiency. Consequently this leads to the ramification that they generate less interest within recent scientific literature concerning electronic elections.

2.2.2. Mix-Nets

An important technique for realizing electronic voting schemes is to use the paradigm of onion routing using mix-nets. On a high level, an e-voting scheme using a mix-net works as follows. Let $V_1, \ldots, V_n$ be a set of voters and $M_1, \ldots, M_m$ be a collection of decrypting mix-servers with public keys $pk_1, \ldots, pk_m$ and $M'_1, \ldots, M'_{m'}$ be a collection of re-encryption mix-servers. We stress that every output of a mix-server is public and posted on some publicly accessible message board.

1. Each voter V_i chooses its votes v_i.
2. Each voter V_i encrypts its vote $c_i^0 := E(v_i) := E(pk_1, \ldots, E(pk_m, v_i) \ldots)$.
3. The k-th mix-server $M_k^{(\prime)}$ takes the values $(c_1^{k-1}, \ldots, c_n^{k-1})$ as input and then:
   a) $M_k^{(\prime)}$ either decrypts or re-encrypts to obtain a new tuple $(c_1^k, \ldots, c_n^k)$.

b) $M_k^{(\prime)}$ secretly permutes the order of the c_i^k and proves that he knows the permutation π.

4. The result is a tuple $(c_1^{m+m'}, \ldots, c_n^{m+m'})$ which can be commonly decrypted in a single step if all proofs were correct.

Most implementations known to us use an ElGamal encryption for the mixes which is presented within the following example.

Example 2 (ElGamal Encryption)**.** *Given a cyclic group $(G, \cdot)$ with generator g of order q (e. g. $G = \mathbb{Z}_q^*$ for prime q) a person B wants to send A an encrypted message $m \in G$ where A has a secret key $x \in \{2, \ldots, q-1\}$. The global setup then is (G, g, q, y) with $y = g^x$ while the ciphertext that B sends to A is $c := (c_1, c_2) := (g^r, my^r)$, where $r \in_R \{2, \ldots, q-1\}$ was chosen at random. The decryption finally is simple: A computes $s := c_1^x = g^{rx}$ and then decrypts by calculating $c_2 \cdot s^{-1} = m$.*

A concrete example for a voting scheme based on re-encrypting mix-nets is Civitas [CCM08] which claims to be the first practical voting scheme realizing incoercibility.

Advantages

The biggest advantage of voting schemes based on mix-nets is their flexibility concerning their type of application. Concretely, the representation of possible votes is not of high importance and especially in elections with many choices (that is approximately 100 or even more) they turn out to be more efficient than other schemes [Lip05]. Another advantage we observe and which has high relevance in practice is the low computational effort needed for a voter to cast a vote which decreases the necessary complexity of software used by a voter. This is due to the fact that a voter does not need perform proofs of knowledge itself. Finally, mix-net based voting schemes become readily universal verifiable due to the fact that every output of a mix-server is publicly accessible.

Disadvantages

Taking a look at the advantages mix-nets seem to give a practical approach for electronic voting schemes. However, they do not perform magic in the sense that they abrogate all the trade-offs. Namely, the advantage that voters have a low computational effort results in a very expensive and complex voting phase since the mix-server have to perform a lot of encryption and decryption processes together with proofs of knowledge that they shuffled the votes honestly. Hence the overall voting process becomes again slow especially in the case where many voters participate within an election. Moreover, to guarantee security it seems to be necessary that all voters have cast their vote *before* tallying begins and this however could lead to a significant delay. Another disadvantage is the problem that in the tallying phase all mix-servers have to be available (e. g. this is the case in Civitas [CCM08]) which makes it hard to achieve availability of the overall voting system since it makes it vulnerable for denial of service attacks on particular authorities. Last, mix-nets also require certain trust assumptions on the mix-server which commonly results in

threshold trust assumptions, i. e. where a specific amount of mix-servers is assumed to be honest.

Conclusions

Albeit mix-net based voting schemes have non-negligible disadvantages, they seem to be a practical approach for electronic voting schemes. This becomes manifest in the observation that a lot of recent e-voting schemes follow this approach. The most important concern however is the high effort for realizing the anonymity since this bottleneck makes many voting schemes based on mix-nets impractical.

2.2.3. Homomorphic Encryption

The third technique used for realizing electronic elections is called *homomorphic encryption* which is the central type of cryptosystem we are concentrated on within this thesis. It is defined as follows.

Definition 1. *A* **homomorphic cryptosystem** *is a public key cryptosystem* (E, D) *with an encryption function* $E : \mathcal{K} \times (\mathcal{M}, +) \longrightarrow (\mathcal{C}, \cdot)$ *that has the additional property that*

$$E(k, m_1) \cdot E(k, m_2) = E(k, m_1 + m_2) \tag{2.1}$$

holds for arbitrary $k \in \mathcal{K}$ *and* $m_1, m_2 \in \mathcal{M}$*, where* $\mathcal{M}$ *and* $\mathcal{C}$ *are arbitrary groups.*

We stress that $\mathcal{M}$ and $\mathcal{C}$ can of course be more general algebraic objects (e. g. rings, modules or fields). However, we require that they are at least common algebraic groups in order to not restrict generality. An example for a homomorphic cryptosystem is the ELGAMAL encryption (Example 2) or the PALLIER encryption presented within the next example, c. f. [Pai99].[1]

Example 3 (PALLIER Encryption)**.** PALLIER *encryption works as follows.*

Setup: *Choose large primes* p, q *of equivalent length, i. e. such that* $p, q \in 1||\{0,1\}^{k-1}$ *for a security parameter* k *and compute* $n = pq$*,* $g := n + 1$*,* $\lambda = \varphi(n)$ *and* $\mu := \varphi(n)^{-1} \bmod n$*. The public key then is* (n, g) *and the private key is* (λ, μ)*.*

Encryption: *Given a message* $m \in \mathbb{Z}_n$*, select* $r \in_R \mathbb{Z}_n^*$ *at random and compute the ciphertext* $c := g^m \cdot r^n \bmod n^2$*.*

Decryption: *Given a ciphertext* $c \in \mathbb{Z}_{n^2}^*$ *compute* $m := \Lambda(c^\lambda \bmod n^2) \cdot \mu \bmod n$*, where* $\Lambda(a) := (a-1)/n$.[2]

[1] We stress that the Pallier encryption is IND-CPA secure under the *decisional composite residuosity* (DCR) assumption, which states that for $N = pq$ and all PPT distinguisher A it holds that $\Pr(A(N, Z) = 1 : \ Z \in_R \mathbb{Z}_{N^2}) \approx \Pr(A(N, Z) = 1 : \ Z' \in_R \mathbb{Z}_N,\ Z := (Z')^N \bmod N^2)$.

[2] The quotient a/b does not mean multiplication with the inverse of b but the common quotient, i. e. the number $a/b := \max\{r \mid a \geq rb\}$. For details on Λ see Lemma 6 in Section 3.4.3.2.

Homomorphic voting then works as follows, given a set of possible votes $\mathcal{V}$, a public key pk, a private key sk and operating with voters $V_1, \ldots, V_n$ and an authority A.

1. Each V_i encrypts its vote $v_i \in \mathcal{V}$, i. e. $x_i := E(pk, v_i)$.
2. Each V_i proves by zero-knowledge, that x_i contains an encrypted valid vote, i. e. that $x_i = E(pk, v_i)$ for some $v_i \in \mathcal{V}$.
3. A collects all casted encrypted votes and calculates their product C.
4. A finally decrypts C giving the result of the election: $\sigma_{\text{Final}} := D(sk, C)$.

It is clear that in such a case the privacy relies heavily on the honesty of A since it may decrypt votes *before* calculating the result. This problem is however usually solved by using homomorphic *threshold* encryption, i. e. where a specific amount of authorities has to cooperate in order to calculate the outcome of the election by sharing the product using a secret share sk_j of the private key sk (for a formal description see Definition 17 in Section 3.2.1).

Advantages

Homomorphic voting has the big advantage that the tallying procedure, i. e. the counting of votes, is very simple since due to the homomorphic property it is possible to calculate the outcome by simply multiplying the ciphertexts without any decryption. Moreover the proofs of correctness of votes could be distributedly performed on different servers and this broads the bottleneck consisting of zero-knowledge proofs. Furthermore it is possible to fractionally evaluate the election *before* all voters have cast their vote which is a big advantage compared to mix-net based voting schemes.

Disadvantages

Homomorphic voting schemes do not abrogate all the trade-offs, either. As mentioned above, it is possible to distribute the effort for verifying the zero-knowledge proofs but they are however still expensive and usual proofs of knowledge turned out to be too cumbersome. This is why modern homomorphic voting schemes use non-interactive zero-knowledge proofs which can be realized efficiently, see e. g. [GS08]. Another critical point is again the threshold trust used to guarantee privacy which makes it necessary to use more voting servers than in mix-net voting schemes. Another disadvantage of homomorphic voting systems is that they are more inflexible than mix-nets based voting schemes since they require special representations of candidates due to the fact that the summation of votes has to reflect the final result in a suitable manner. Moreover, these schemes turn out to be inefficient in elections with many choices since the costs of vote verification depends mostly on the number of candidates [PB11, p. 383]. Finally, especially additive homomorphic voting schemes have the drawback that the key distribution in their case is mostly very complex since they use factorization as a trapdoor (e. g. this is the case in ElGamal and Pallier encryption), c. f. [PB11, p. 381–382].

Conclusions

Homomorphic voting schemes have the big advantage that the tallying process is very fast. Namely, no decryption has to be carried out until the decryption of the overall outcome, since tallying is done via the homomorphism property of E. However, also homomorphic schemes have several drawbacks which are mostly bottlenecks in inefficiency but in our opinion it is conceivable that homomorphic voting schemes are more efficient than mix-net based protocols though. Of course, this competition is still actual in modern research and it is an open problem which approach leads to efficient and practical voting schemes.

3. Evaluating in the UC Framework

As we have mentioned within the introduction and Section 2.1 it is quite difficult to find a consensus in the literature which security requirements are necessary for an e-voting scheme to be "secure". As time has processed a lot of e-voting protocols were published, most of which defining their own security terms and only rarely providing security proofs, see for example [JCJ05, BT94, CCM08, CGS97, DJ01, BFP+01, DGS02] among others. For that reasons Groth [Gro04] proposed to evaluate security within the *Universal Composability* (UC) framework of Canetti [Can01]. The advantage in doing so is that the UC framework itself already provides us with a strong simulation based security definition and, additionally, supports concurrent execution of protocols even in asynchronous networks. Namely, it comes along with an universal composition theorem. This composition theorem in turn has the consequence that, roughly speaking, every secure protocol π is, composed with any other secure protocol ρ, again secure in the sense of the UC framework. As we think that this is a possibly fruitful approach we give a sketchy introduction of the UC framework in this chapter with the intention to present the necessary theoretical background to understand the security proofs made in the sections thereafter. Namely it turns out that it is indeed possible to define a UC-secure protocol, i. e. realizing a protocol π_{VOTING} that securely realizes an *ideal functionality* $\mathcal{F}_{\text{VOTING}}$, which models a lot of considerable ideal functionalities for an e-voting protocol. The gist then is that many of the security requirements presented in Section 2.1 easily follow as corollaries. Moreover, if it would be possible to agree on an ideal e-voting functionality $\mathcal{F}_{\text{VOTING}}$ which is suitable, we would gain a good approach for comparing e-voting protocols by simply proving if they realize this functionality. Unfortunately this is not as simple as it sounds but we believe that this is possible though.

3.1. The UC Framework

This section is intended to give a brief introduction into the Universal Composability (UC) framework of Canetti [Can01] (we stress that, to this end, all page references refer to the full version of the paper). We begin with preliminary definitions of fundamental objects and concepts. Thereafter we will define the basic UC framework and explain what it means for a protocol π to be secure in the UC framework and this will lead to the definition of a *secure realization* of a protocol. Note that our presentation here is just an excerpt meaning that we present the theoretical fundamentals we need for further investigations only. For a comprehensive and complete overview of the framework we refer the reader to the original paper [Can01]. We begin with the definition of an *interactive Turing machine* $\mathcal{M}$ which is the central object within the UC framework as it models parties participating in an execution of a protocol π, c. f. [Can01, pp. 27].

Definition 2. *An* **interactive Turing machine** *(ITM)* $\mathcal{M}$ *is a* 9*-tape Turing machine with the following tapes.*

1. *An EW* `identity` *tape.*
2. *An EW* `security parameter` *tape.*
3. *An EW* `input` *tape.*
4. *An EW incoming communication tape* `IC`*.*
5. *An EW subroutine output tape* `SO`*.*
6. *An* `output` *tape.*
7. *A* `random` *tape.*
8. *A read and write one-bit* `activation` *tape.*
9. *A read and write* `work` *tape.*

Thereby, a tape of a Turing machine is called **externally writable** *(EW), if it may be written to by other ITMs and its writing head can only move in a single direction. The content of the* `identity` *tape is interpreted as two strings: the session identity* SID *and the unique party identity* PID *while the values on the* `IC` *tape are called* `messages`*.*

In order to define what we understand under an *execution of a protocol* π we need to define what we understand under a *system of ITMs* and how we execute such a system.

Definition 3. *Let* M *be an ITM. A* **configuration** $\mathcal{C}_M := (C_M, s_M, \mathcal{T}_M, \mathcal{H}_M)$ *of* M *is a 4-tuple consisting of the code* C_M*, the current control state* s_M*, the content* $\mathcal{T}_M$ *of all tapes of* M *and the head positions* $\mathcal{H}_M$*. A configuration* $\mathcal{C}_M$ *is said to be* **active** *(or* **inactive***) if the content of* `activation` *is set to 1 (or 0). An* **instance** $\mu_M = (C_M, \mathrm{id})$ *of* M *consists of its code* C_M *and an* $\mathrm{id} \in \{0,1\}^*$*. Finally,* $\mathcal{C}_M$ *is a* **configuration of instance** $\mu_M = (C_M, \mathrm{id})$*, if the codes are the same and the content of the* `identity` *tape in* $\mathcal{C}_M$ *is equal to* id*.*

Let M *and* M' *be ITMs and* $\mu_M = (C_M, \mathrm{id})$*,* $\mu_{M'} = (C_{M'}, \mathrm{id}')$ *some instances, respectively.* M *may now include an* **external-write** *instruction to* M' *consisting of* μ_M*,* $\mu_{M'}$*, the target tape to be written to (either* `IC`*,* `SO` *or* `input`*) and some data* $d \in \{0,1\}^*$ *to be written. Depending on the tape that is written to, there exist some intricacies, c. f. [Can01, p. 29], but basically the data* $d \in \{0,1\}^*$ *is simply written onto the target tape of* $\mu_{M'}$*.*

Finally, an ITM instance μ_M *of an ITM* M *may* **invoke** *another instance* $\mu_{M'}$ *of an ITM* M' *by specifying an external-write instruction to* $\mu_{M'} = (C_{M'}, \mathrm{id}')$*. This is done by generating a new configuration* $C_{M'}$ *of* M'*, where* id' *is written on the* `identity` *tape,* 1^k *is written on the* `security parameter` *tape and a random string is written onto the* `random` *tape. An* **external-write** *instruction from* μ_M *to* $\mu_{M'}$ *as above can only be carried out if* $\mu_{M'}$ *has been invoked, i. e. if there already exists a configuration of instance* $\mu_{M'}$*.*

Definition 4. *A* **system of ITMs** $\mathcal{S} = (\mathcal{I}, C)$ *consists of an initial ITM* $\mathcal{I}$ *together with a control function* $C : \{0,1\}^* \longrightarrow \{\texttt{allow}, \texttt{disallow}\}$. *This tuple becomes a system of ITMs due to the fact that* $\mathcal{I}$ *may invoke other ITM instances (c. f. Definition 3).*

Thereby, the control function C *determines if an external-write instruction by an ITM is allowed or not and the sequence given to* C *as input is the sequence of external-write instructions in the system so far.*

Whenever an ITM writes onto another's `SO` *or* `input` *tape, we say that it, respectively,* passes output or input *to it. Moreover, if it writes onto the* `IC` *tape of another ITM we say that it* sends a message.

The output of $\mathcal{S}$ *is the content of the* `output` *tape of* $\mathcal{I}$ *when it halts. Given a security parameter* $k \in \mathbb{N}$ *and some input* $x \in \{0,1\}^*$, *the system* $\mathcal{S}$ *is executed in the following way:* x *is written on the* `input` *tape,* 1^k *is written on the* `security parameter` *tape and some random value* $r \in \{0,1\}^*$ *is written on the* `random` *tape of* $\mathcal{I}$. *The ITM* $\mathcal{I}$ *is then invoked at first with identity* 0 *and* $\mathcal{I}$ *may now invoke other ITM instances using external-write instructions, whereby at most one ITM instance can be active and which again may invoke or communicate with other ITM instances, if allowed by the control function* C.

Before we get concrete in defining the protocol execution in the UC framework, we will need to define *probabilistic polynomial time* ITMs which will be the objects needed for further security definitions.

Definition 5. *Let* $p : \mathbb{N} \longrightarrow \mathbb{N}$ *be an arbitrary polynomial. We say that an ITM* $\mathcal{M}$ *is* **locally** p**-bounded** *if, at any point during the execution of* $\mathcal{M}$, *the overall number of computational steps taken by* $\mathcal{M}$ *so far is at most* $p(n)$, *where*

$$n = k + n_I - n_O - k \cdot n_N.$$

Here, $k \in \mathbb{N}$ *is the security parameter,* n_I *is the overall number of bits written so far on the* `input` *tape of* $\mathcal{M}$, n_O *is the number of bits written by* $\mathcal{M}$ *so far to* `input` *tapes of other ITMs and* n_N *is the overall number of these ITMs.*

If, additionally, $\mathcal{M}$ *does only communicate with locally* p*-bounded ITMs (or does not even communicate at all), then* $\mathcal{M}$ *is said to be* p**-bounded**. *Furthermore,* $\mathcal{M}$ *is* **PPT**, *if there exists a polynomial* p *such that* $\mathcal{M}$ *is* p*-bounded.*

A multiparty protocol π is now modeled as an ITM with a unique SID and an *instance of* π is a set of ITM instances (the *parties*). Concretely, each party that participates within the protocol is defined as a single ITM with the SID *sid* of the protocol instance and a unique PID *pid*.

The *execution of* π is a system of ITMs (c. f. Definition 4), represented through π, a unique initial ITM $\mathcal{Z}$ (the *environment*), an additional ITM $\mathcal{A}$ (the *adversary*) and a specific control function $C^{\pi,\mathcal{A}}_{\text{EXEC}}$, depending on π and $\mathcal{A}$. Note that, due to Definition 4, it is possible for invoked parties of π (i. e. invoked instances of π with special PID) to invoke other instances, too. These instances however are then called *sub-parties* of π since they were invoked by parties of π. Concretely, an execution is defined as follows.

Definition 6. *An* **execution of a protocol** π *in the UC framework is defined as the system* $(\mathcal{Z}, C^{\pi,\mathcal{A}}_{\text{EXEC}})$ *of ITMs with*

- *a PPT ITM* $\mathcal{Z}$*, called the* **environment**,
- *a PPT ITM* $\mathcal{A}$*, called the* **adversary**, *and*
- *a PPT ITM* π *for the given protocol.*

Here, the control function $C^{\pi,\mathcal{A}}_{\text{EXEC}}$ *is defined as follows. Initially,* $\mathcal{Z}$ *is invoked with some initial input* $z \in \{0,1\}^*$ *and a security parameter* $k \in \mathbb{N}$*. Then:*

(C1) The first ITM invoked by $\mathcal{Z}$ *is the adversary* $\mathcal{A}$*. All other ITMs invoked by* $\mathcal{Z}$ *have the code of* π *and the same* SID*.*

(C2) When the adversary $\mathcal{A}$ *is activated, it can pass output to* $\mathcal{Z}$ *and additionally perform one of the following activities.*

(C2a) $\mathcal{A}$ *can deliver a message* m *to a (sub-)party.*

(C2b) $\mathcal{A}$ *can corrupt a (sub-)party if it is instructed by* $\mathcal{Z}$ *to do so via a corresponding* corrupt *message from* $\mathcal{Z}$*.*

(C3) If a (sub-)party of π *is activated (by* $\mathcal{Z}, \mathcal{A}$ *or another (sub-)party), then it can send messages to* $\mathcal{A}$ *and pass input or output to other (sub-)parties with the same* SID*. Additionally, parties of* π *can pass output to* $\mathcal{Z}$*.*

We want to point out some crucial but subtle details following from the definition of the control function $C^{\pi,\mathcal{A}}_{\text{EXEC}}$ in Definition 6.

Remark 1.
1. *By property (C2a),* $\mathcal{A}$ *can only send messages to sub-parties but has neither access to their* `input` *tape nor their* `SO` *tape (i. e. the adversary can not pass input or output to (sub-)parties).*
2. *By property (C3), neither* $\mathcal{Z}$ *can pass input to sub-parties nor can sub-parties pass output to* $\mathcal{Z}$ *directly.*
3. *By property (C3), a (sub-)party can only send messages to* $\mathcal{A}$ *but not to other (sub-)parties.*

The reaction to corruption messages of parties or sub-parties is thereby modeled within the protocol and thus not part of the definition. Usually, parties will send their entire state to the adversary and are henceforward controlled by the adversary (this is known as the Byzantine corruption model).

The following example illustrates a very simple execution of the e-voting protocol using homomorphic encryption.

Example 4. *We give a brief example of an execution. Consider the simple protocol for an electronic election using homomorphic encryption from Section 2.2.3 following Example 3. This protocol, denoted with π_V, participates with N Voters $V_1, \ldots, V_N$ and a trusted party A. We stress that π_V is far away from being practically useful but it is simple enough to present an execution of it within the UC framework. In particular, we use the* Pallier *encryption from Example 3 to encrypt votes. The example environment $\mathcal{Z}$ will simply lead through the execution process of the protocol. To do so, $\mathcal{Z}$ lets each voter cast its vote and, additionally, adaptively corrupts a single voter at random which has the effect that the vote of this voter is negated. The example shows in particular, how the different tapes of an ITM are used in a concrete protocol execution and how instances of ITMs are invoked.*

ITMs within the execution

Let $\mathcal{V} := \mathbb{Z}_2 = \{0,1\}$ denote the set of possible votes and consider the following informally described ITMs in the network.

$\mathcal{Z}$: The environment $\mathcal{Z}$ gets as input the N-tuple $(v_1, \ldots, v_N) \in \mathcal{V}^N$ consisting of the votes for each voter and a security parameter $k \in \mathbb{N}$. $\mathcal{Z}$ then proceeds as follows:

- *generate an* SID *sid for the current execution;*
- *invoke $\mathcal{A}$;*
- *write v_i onto the* `input` *tape of V_i;*
- *send a* **generate-keys***-message to $\mathcal{A}$;*
- *send a* **corrupt***-message for party V_ℓ to $\mathcal{A}$ for some $\ell_R \in \{1, \ldots, N\}$;*
- *send a* **vote***-message to each V_i, $i \neq \ell$, to let them vote;*
- *send a* **get-result***-message to A;*
- *when receiving σ_{Final} from A then halt with output 1.*

$\mathcal{A}$: The adversary $\mathcal{A}$ proceeds as follows:

- *when receiving a* **generate-keys***-message from $\mathcal{Z}$ then forward it to A;*
- *when receiving a* **corrupt***-message for party V_ℓ from $\mathcal{Z}$ forward it to V_ℓ;*
- *when receiving the complete state of V_ℓ, then vote on behalf of V_ℓ with $\neg v_\ell$.*

V_i: When receiving a **vote***-message, each uncorrupted voter reads $r \in \mathbb{Z}_n^*$ from its* `random` *tape and writes its encrypted vote $x_i := g^{v_i} r^n \bmod n^2$ onto the* `SO` *tape of A. If V_i receives the* **corrupt***-message, then it hands out its entire configuration to $\mathcal{A}$.*

A: The authority A proceeds as follows:

- *when receiving a* **generate-keys***-message from $\mathcal{A}$, then choose large primes $p, q \in 1||\{0,1\}^{k-1}$ and compute $n := pq$, $g := n + 1$, $\lambda = \varphi(n)$ and $\mu := \varphi^{-1}(n) \bmod n$ due to the* Pallier *encryption;*
  write $pk := (n, g)$ onto the `input` *tapes of $V_1, \ldots, V_N$ and save $sk := (\lambda, \mu)$ as secret key;*

- *when receiving a* **get-result***-message from $\mathcal{Z}$, multiply all encrypted votes on the* `SO` *tape and write the decrypted product σ_{FINAL} to the* `SO` *tape of $\mathcal{Z}$.*

Step-by-step description of the execution

We will now describe how the execution of π due to Definition 6 is generated.

1. $\mathcal{Z}$ **is active**. *By definition, the first ITM that is invoked, is the environment $\mathcal{Z}$ with external input $(v_1, \ldots, v_N)$ and $k \in \mathbb{N}$. Next, $\mathcal{Z}$ chooses a* SID *sid which specifies the current execution process, and then invokes $\mathcal{A}$, i. e. it writes $k \in \mathbb{N}$ onto the* `security parameter` *tape of $\mathcal{A}$ and proceeds due to the description of the invocation in Definition 3.*

2. $\mathcal{A}$ **is active**. *The* `activation` *tape of $\mathcal{A}$ changes to 0 as $\mathcal{A}$ does not perform any action.*

3. $\mathcal{Z}$ **is active**. *$\mathcal{Z}$ writes each v_i onto the* `input` *tape of V_i. This implies, that every voter V_i is invoked by $\mathcal{Z}$, i. e. new instances of π_V with special input v_i and* PID *pid_i are generated.*

4. A_i **is active**. *The* `activation` *tape of A_i changes to 0 as A_i does not perform any action.*

5. $\mathcal{Z}$ **is active**. *$\mathcal{Z}$ sends a* **generate-keys***-message to the adversary which forwards it to A. This in turn implies again an implicit invocation of A, i. e. an instance of the ITM π_V with special* PID *pid_A is generated.*

6. A **is active**. *After the invokation of A, it is active and generates the data for the* PALLIER *encryption. It writes pk to the* `input` *tape of each V_i. The* `activation` *tape of A then changes to 0.*

7. $\mathcal{Z}$ **is active**. *$\mathcal{Z}$ sends a* **corrupt***-message for some V_ℓ to $\mathcal{A}$.*

8. $\mathcal{A}$ **is active**. *$\mathcal{A}$ forwards the* **corrupt***-message to V_ℓ and learns the entire states of the affected voter. It then writes $x_\ell := g^{\neg v_\ell} r^n \bmod n^2$ onto the* `SO` *tape of A on behalf of V_ℓ. The* `activation` *tape of $\mathcal{A}$ then changes to 0.*

9. $\mathcal{Z}$ **is active**. *$\mathcal{Z}$ sends a* **vote***-message to every voter V_i, $i \neq \ell$.*

10. V_i **is active**. *V_i calculates its encrypted vote x_i due to its specification and writes x_i onto the* `SO` *tape of A.*

11. $\mathcal{Z}$ **is active**. *$\mathcal{Z}$ sends a* **get-result***-message to A.*

12. A **is active**. *A calculates σ_{FINAL} and acts due to its specification.*

13. $\mathcal{Z}$ **is active**. *Having the final result on the* `SO` *tape, $\mathcal{Z}$ halts with output 1.*

This completes the example execution.

By simplicity, we restrict to environments $\mathcal{Z}$ that output a single bit and fix a notation:

Definition 7. *Given some input $z \in \{0,1\}^*$ and security parameter $k \in \mathbb{N}$, we define $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z) \in \{0,1\}$ to be the output of the system $(\mathcal{Z}, C^{\pi,\mathcal{A}}_{\mathrm{EXEC}})$ (c.f. Definition 6). Naturally, this binary variable induces a binary distribution ensemble $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ defined through*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} := \bigcup_{k\in\mathbb{N}} \bigcup_{z\in\{0,1\}^*} \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z).$$

Before we define, how a protocol π can UC-emulate another protocol ρ, we need a last technical definition, defining the indistinguishability of two binary distribution ensembles. The definition of this type of emulation is crucial for our further investigations since it is essential in the definition of *secure realizations of protocols in the UC framework*, being the central concept of interest within the rest of this thesis.

Definition 8. *Let $X := \{X(k,a)\}_{a\in\{0,1\}^*,k\in\mathbb{N}}$ and $Y := \{Y(k,a)\}_{a\in\{0,1\}^*,k\in\mathbb{N}}$ denote two binary distribution ensembles, i. e. we have $X(k,a) \in \{0,1\}$ and $Y(k,a) \in \{0,1\}$ being two binary random variables for all $a \in \{0,1\}^*$ and $k \in \mathbb{N}$. We call X and Y* **indistinguishable** *(denoted with $X \approx Y$), if for any $c,d \in \mathbb{N}$ there exists a $k_0 \in \mathbb{N}$ such that for all $k > k_0$ and all $a \in \bigcup_{\kappa\leq k^d}\{0,1\}^\kappa$ it holds*

$$|\Pr(X(k,a)=1) - \Pr(Y(k,a)=1)| < k^{-c}.$$

Remark 2. *The above definition is equivalent to simply requiring that for all $d \in \mathbb{N}$ there exists a negligible function μ_d such that*

$$|\Pr(X(k,a)=1) - \Pr(Y(k,a)=1)| \leq \mu_d(k)$$

for all $k \in \mathbb{N}$ and all $a \in \bigcup_{\kappa\leq k^d}\{0,1\}^\kappa$. Thereby, we call a function $\mu : \mathbb{N} \longrightarrow [0,1]$ **negligible***, if for all $c > 0$ it holds $\mu(k) = O(k^{-c})$ for sufficiently large $k \in \mathbb{N}$.*

We are now in the position to define the emulation of a protocol through another.

Definition 9. *Let π and ρ be PPT protocols. We say that π* **UC-emulates** *ρ, if for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

This concept of emulation is well-known as it represents the idea to compare the execution of a protocol π in the "real world" with an execution of another protocol ρ in an "ideal world", where a simulator $\mathcal{S}$ tries to adapt an adversary $\mathcal{A}$ "as good as possible". However, the gist here is that the environment $\mathcal{Z}$ acts as a distinguisher between the interactions in both protocol emulations. The final idea is now to replace the protocol ρ in Definition 9 with a special type of protocol representing the "ideal world" more specifically. Namely, an *ideal protocol* is substituted for ρ, where the ideal protocol in turn simulates an *ideal*

functionality which finally represents the ideal and thus expected behavior of the protocol π. Once we have defined what we understand under an ideal functionality $\mathcal{F}$ and the related ideal protocol $\text{IDEAL}_{\mathcal{F}}$, we are able to formulate what is meant, when a multi-party protocol π securely realizes an ideal functionality $\mathcal{F}$ within the UC framework. This finally completes our general approach to define a simulation based security concept that enables us to analyze arbitrary protocols concerning their security in the sense just mentioned. We want to point out, that this concept was not created from scratch by CANETTI. Namely, seminal work was done by GOLDREICH, MICALI and WIGDERSON within [GMW87].

Definition 10. *An* **ideal functionality** $\mathcal{F}$ *is an ITM with the following additional conventions which are mostly not allowed to conventional parties participating within the protocol emulation (c. f. Definition 6).*

(I1) The PID *of* $\mathcal{F}$ *is set to* $\perp$.

(I2) The `input` *tape of* $\mathcal{F}$ *can be written to by multiple ITMs (with the same* SID*).*

(I3) The `IC` *tape of* $\mathcal{F}$ *is used to communicate with the adversary* $\mathcal{A}$.

(I4) $\mathcal{F}$ *can write to* `SO` *tapes of multiple ITMs in the system.*

The idea behind this definition is, that an ideal functionality should model a subroutine machine which is jointly used by other ITMs in the system. The protocol corresponding to an ideal functionality is now defined as follows.

Definition 11. *The* **ideal protocol** $\text{IDEAL}_{\mathcal{F}}$ *corresponding to an ideal functionality* $\mathcal{F}$ *is defined as the following protocol.*

Input: *dummy parties* $\widetilde{P}_1, \ldots, \widetilde{P}_n$ *with* SID *sid and* PIDs $pid_1, \ldots, pid_n$.

- *Ignore messages of* $\mathcal{A}$*;*
- **if** *party* $\widetilde{P}_i$ *is activated with input* v_i **then** *write* v_i *on the* `input` *tape of the instance of* $\mathcal{F}$ *with* SID *sid ;*
- **if** *party* $\widetilde{P}_i$ *receives* u_i *on its* `SO` *tape* **then** *write* u_i *on the* `SO` *tape of* $\mathcal{Z}$ *;*

Figure 1: Protocol $\text{IDEAL}_{\mathcal{F}}$

Let us briefly recap this definition. First, the ideal protocol obviously does nothing else than modeling the functionalities of $\mathcal{F}$ into a multi-party frame by simply relaying inputs and outputs from $\mathcal{Z}$ to an instance of the ideal functionality $\mathcal{F}$ with the same SID and vice versa. Secondly, it ignores any message from the adversary $\mathcal{A}$ in order to abandon corruption responses to the ideal functionality itself; in particular dummy parties ignore possible corruption messages. With the usage of this definition, we finally obtain a new type of protocol emulation. Namely we are in the position to define what is meant under an UC-secure realization of a protocol π that emulates a desired functionality $\mathcal{F}$.

Definition 12. *Let π be a PPT protocol and $\mathcal{F}$ an ideal functionality. We say that π* **UC-realizes** *$\mathcal{F}$, if π UC-emulates* $\text{IDEAL}_{\mathcal{F}}$*, i. e. for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{S}$ such that*

$$\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}} \approx \text{EXEC}_{\text{IDEAL}_{\mathcal{F}},\mathcal{S},\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

Of course, this definition is crucial for the forthcoming parts of this thesis since it describes the type of security we investigate. Concretely it will be a main purpose to present an ideal functionality $\mathcal{F}_{\text{VOTING}}$ for an e-voting protocol from [Gro04] which is a first step forward to a general definition of a *secure* e-voting protocol. As we will see, however, this functionality does not cover all desired properties mentioned in Section 2.1, but it is a first good attempt towards a unification of theoretical investigations of e-voting protocols. The main gist is, that the ideal functionality $\mathcal{F}_{\text{VOTING}}$ is in fact UC-realizable through a protocol π_{VOTING}. We stress that this is not trivial, e. g. it can be shown that the zero-knowledge functionality $\mathcal{F}_{\text{ZK}}$ is, among others, never UC-realizable in the two-party case, c. f. [Can01, Section 7.3]. (For the sake of completeness we stress that with the aid of a *common reference string* (CRS) this is possible though.)

As we will see, it is often useful to encapsulate different functionalities from a concrete protocol π in the real world. This encapsulation is again realized via some ideal functionality $\mathcal{F}$ which is executed in the "real world", i. e. within the emulation of π. This is done by simply integrating the ideal protocol $\text{IDEAL}_{\mathcal{F}}$ into the emulation of π. We thus obtain some protocol "hybrid" which is formally defined in the following.

Definition 13. *An* **$\mathcal{F}$-hybrid protocol** *π is a protocol that additionally includes subroutine calls to the ideal protocol* $\text{IDEAL}_{\mathcal{F}}$ *(i. e. it invokes dummy parties of* $\text{IDEAL}_{\mathcal{F}}$ *which in turn invoke instances of $\mathcal{F}$).*

For the sake of completeness we finish our presentation of the UC framework with the definition of the *universal composition of two protocols* π and ρ along with a citation of the universal composition theorem [Can01, Theorem 14].

Definition 14. *Let π, ρ and ρ' be protocols, where π might make subroutine calls to instances of the protocol ρ. Then $\pi^{\rho'/\rho}$ is defined as being identical to the protocol π with the following two modifications:*

(M1) Whenever π contains an instruction to pass input z to an ITM running ρ with `identity` (sid, pid)*, then $\pi^{\rho'/\rho}$ passes z to an ITM running ρ' with* `identity` (sid, pid)*.*

(M2) Whenever $\pi^{\rho'/\rho}$ receives an output from an ITM running ρ' with `identity` (sid, pid)*, then it proceeds as π when receiving output from an ITM running ρ with* `identity` (sid, pid)*.*

Alternatively, when ρ is an ideal protocol $\text{IDEAL}_{\mathcal{F}}$ *for an ideal functionality $\mathcal{F}$, we define* $\pi^{\rho'/\mathcal{F}} := \pi^{\rho'/\,\text{IDEAL}_{\mathcal{F}}}$.

Note that instances of $\pi^{\rho'/\rho}$ are again PPT ITMs if so are π, ρ and ρ' [Can01, p. 53]. Finally, in order to formulate the universal composition theorem we need to define a last special type of protocols.

Definition 15. *Consider a protocol π and a protocol ρ where π might make subroutine calls to instances of the protocol ρ. We say that an instance $\rho_{(sid,pid)}$ with* `identity` *(sid, pid) of ρ within an execution of π is* **subroutine respecting**, *if*

(i) no sub-party of $\rho_{(sid,pid)}$ receives inputs or outputs from other (sub-)parties except those of $\rho_{(sid,pid)}$.

(ii) no sub-party of $\rho_{(sid,pid)}$ passes inputs or outputs to other (sub-)parties except those of $\rho_{(sid,pid)}$.

A protocol ρ is itself **subroutine respecting**, *if every instance of ρ, in any execution of any protocol π that makes subroutine calls to this instance of ρ, is subroutine respecting as defined above.*

The universal composition theorem [Can01, Theorem 14] now states the following.

Theorem 1 (Universal Composition)**.** *Let π, ρ and ρ' be PPT multi-party protocols such that ρ UC-emulates ρ' and both ρ and ρ' are subroutine respecting. Then protocol $\pi^{\rho'/\rho}$ UC-emulates protocol π.*

The theorem has important consequences: once we have found a protocol ρ that might securely realizes an ideal functionality $\mathcal{F}$, then we can run this protocol as subprotocol within some protocol π that makes subroutine calls to ρ. Theorem 1 now implies that in this case $\pi^{\rho/\mathcal{F}}$ UC-emulates π (where ρ' in the theorem is replaced by $\text{IDEAL}_{\mathcal{F}}$). In other words, we can use our secure protocol ρ within presumably more complex protocols π that make subroutine calls to $\mathcal{F}$ without any consequences on the security. Another corollary from Theorem 1 is the following: If π is a $\mathcal{F}$-hybrid protocol that UC-realizes an ideal functionality $\mathcal{G}$ and ρ UC-realizes $\mathcal{F}$ then $\pi^{\rho/\mathcal{F}}$ UC-realizes $\mathcal{G}$ [Can01, Corollary 16]. The last conclusion is important as it has the consequence that we might realize some desired functionality $\mathcal{F}$ (e. g. $\mathcal{F}_{\text{VOTING}}$) using hybrid protocols. This has the advantage that we can rely on forthcoming constructions of e-voting protocols that UC-realize $\mathcal{F}_{\text{VOTING}}$ on ideal functionalities, which go beyond the scope of a first birds eye's view. Of course we will make usage of this considerations within Section 3.3.

3.2. Cryptographic Preliminaries

The current section is devoted to the presentation of an e-voting protocol that UC-realizes an ideal functionality $\mathcal{F}_{\text{VOTING}}$ of electronic remote voting. The protocol on hand was constructed by GROTH [Gro04] and we will give a detailed overview within this chapter. Concretely we present the protocol within Section 3.3 along with the necessary security proofs while this preliminary section is intended to give an introduction into underlying

cryptographic tools. Namely the protocol in Section 3.3 is based on homomorphic threshold encryption, which is a special type of public key encryption, where a specific number of authorities has to cooperate in order to decrypt messages. In particular the authorities can perform calculations with ciphertexts without any knowledge on the corresponding plaintexts. Moreover we introduce a special type of non-interactive zero knowledge proofs of knowledge (NIZK) which are used within the protocol to convince authorities that casted votes are valid.

3.2.1. Homomorphic Threshold Encryption

As described in Section 2.2.3, a homomorphic cryptosystem is a cryptosystem with an encryption function $E : \mathcal{K} \times (\mathcal{M}, +) \longrightarrow (\mathcal{C}, \cdot)$ that has the additional property that

$$E(k, m_1) \cdot E(k, m_2) = E(k, m_1 + m_2) \tag{3.1}$$

holds for arbitrary $k \in \mathcal{K}$ and $m_1, m_2 \in \mathcal{M}$. Here, $\mathcal{M}$ and $\mathcal{C}$ are groups or even more particular algebraic objects. If we write $\mathcal{K}$ and $\mathcal{M}$ additively and $\mathcal{C}$ multiplicatively, we speak of *additive homomorphic encryption*. However, we may also consider the case where $\mathcal{C}$ is a multiplicative group as well and then speak of *multiplicative homomorphic encryption*. In the multiplicative case, we commonly assume that there exists a unique factorization into irreducible elements, which is the case if $\mathcal{M}$ is at least a unique factorization domain. The intention behind the usage of this type of cryptography was already discussed in Section 2.2.3. However, the special type of cryptosystems that we will use within the protocol presented in Section 3.3, is a little different. Namely we consider (t, n)-*threshold systems* with an encryption function E that is homomorphic in the sense of (3.1). We define such a (t, n)-*threshold system* as follows.

Definition 16. *A* (t, n)**-threshold system** *is a public key cryptosystem* (E, D) *acting with* n *authorities* $A_1, \ldots, A_n$ *in order to decrypt messages, when at least* t *of them cooperate. Concretely, it is designed as follows.*

- *A key generation algorithm* K *generates a* **public key** pk*, public* **verification keys** $vk_1, \ldots, vk_n$ *and a* **secret key** sk*. Here,* sk *is shared between* $A_1, \ldots, A_n$*, i. e. each authority* A_i *holds a* **secret share** sk_i*,* $i = 1, \ldots, n$*, of* sk*.*

- *To encrypt a message* m*, the encryption function* E *of the cryptosystem is run with the public key* pk*.*

- *To decrypt a ciphertext* c*, each* A_i*,* $i = 1, \ldots, n$*, uses its secret share* sk_i *to generate a* **decryption share** ds_i*, which is proven by zero-knowledge to be correct by using its verification key* vk_i*. If any* A_i *has received at least* t *decryption shares* ds_j*, it can combine them and decrypt* c*. The decryption function* $D : \mathcal{C} \times \mathcal{C}^t \longrightarrow \mathcal{M}$ *in such a cryptosystem is called* **combining function**. *It takes as input the ciphertext* c *together with* t *decryption shares of it and produces as output the plaintext to* c*.*

In addition, we require that the cryptosystem has the following properties.

Semantic security: *The cryptosystem has to be semantically secure, that is, it must be infeasible for any PPT adversary $\mathcal{A}$ to derive significant information about the plaintext when given only its ciphertext, i. e. it holds for all $m, m' \in \mathcal{M}$ that $|\Pr(pk \in_R \mathcal{K} : \mathcal{A}(E(pk, m)) = 1) - \Pr(pk \in_R \mathcal{K} : \mathcal{A}(E(pk, m')) = 1)| \leq \frac{1}{3}$.*

Errorless decryption: *The key generator K selects with overwhelming probability keys such that there is probability 1 for the decryption to yield the message encrypted.*

Simulatability: *There exists a simulator S that, when taking as input a ciphertext c, a message m and the secret shares of $t-1$ authorities, can produce simulated decryption shares for the remaining authorities such that c decrypts to m. Thereby, the simulated decryption shares must be – even with knowledge of the $t-1$ secret shares – indistinguishable from the real decryption shares.*

Let us give a short discussion of the previous definition. First, the presented explanation of the required semantic security only considers the case where an adversary may generate ciphertexts using the public key and plaintexts of its choice (chosen plaintext attacks, abbreviated as CPA). Thus we require our cryptosystem to be IND-CPA (indistinguishable under CPA) since a stronger security requirement of the cryptosystem used to encrypt votes – such as IND-CCA and IND-CCA2 – would contradict our intention to use a *homomorphic* cryptosystem. The reason is that these systems are obviously malleable[1] by design. Secondly, the simulatability property guarantees that, even with knowledge of $t-1$ secret shares, no adversary is able to gain any information about the ciphertext encrypted under the public key of the (t, n)-threshold cryptosystem. We point out that this simulation paradigm is crucial for forthcoming security proofs. Last, note that the key generation algorithm K is not necessarily executed by a trusted authority so that indeed no single authority can decrypt messages encrypted under the public key pk.

Definition 17. *A cryptosystem with the properties of Definition 1 and Definition 16 is called* **homomorphic (t, n)-threshold cryptosystem**.

An example for such a threshold cryptosystem can be found in Section 3.4.3.2 which uses a threshold version of PALLIER encryption that was introduced in Section 2.2.3. Also within this section, we mentioned that homomorphic cryptosystems are malleable by design. The problem is that a corrupt voter may cast an ineligible vote which modifies the outcome (i. e. the result after tallying the casted votes using the homomorphic property) in an illegal manner. As an example consider a voting system where a voter might only cast 0 or 1 as vote. Without proving that he has casted a value of $\{0, 1\}$, he might also cast a (-5)-vote which then decreases the result by the homomorphic property of

[1] Let $R \subset \mathcal{M} \times \mathcal{M}$ be some relation. A public key cryptosystem with encryption function $E : \mathcal{M} \longrightarrow \mathcal{C}$ is said to be *non-malleable*, if for all PPT adversaries $\mathcal{A}$ that, given a plaintext $m \in \mathcal{M}$ and $c = E_{pk}(m)$ as input, outputs a ciphertext $c' \leftarrow \mathcal{A}(m, c, R)$ with $c' = E_{pk}(m')$ for some $m' \in \mathcal{M}$, it exists an expected PPT emulator $E_{\mathcal{A}}$ such that $\Pr((m, m') \in R : c' \leftarrow \mathcal{A}(m, c, R)) \approx \Pr((m, m'') \in R : c'' \leftarrow E_{\mathcal{A}}(R))$.

the cryptosystem without being noticed by anyone. Thus, instead of merely casting its encrypted vote, we will force the voter to, additionally, send a proof that he has casted a *valid* vote without revealing this. Namely, this is done using a special class of zero-knowledge proofs which we present within the next section.

3.2.2. Sigma-Protocols and Non-interactive Zero-Knowledge Proofs

Since conventional zero-knowledge proofs are too cumbersome and inefficient for practical uses, Groth [Gro04, p. 3] proposes to use Sigma-protocols (or briefly Σ-protocols) which are transformed into non-interactive zero-knowledge proofs by using the Fiat-Shamir heuristic, i. e. they introduce a random oracle which outputs challenges. A Σ-protocol is a 3-move honest verifier zero-knowledge proof introduced by Cramer [CDS94] which in turn is a special proof of knowledge [BG93]. Formally, we define such a protocol as follows.

Definition 18. *A* **proof of knowledge** *for a relation $R = \{(x, w) \mid x \in L,\ w \in W(x)\}$, where L is a* NP*-language and $W(x) \subset \{0,1\}^*$ for $x \in L$ is the set of witnesses, is a two-party protocol running with a PPT prover P and a PPT verifier V with the following two properties. As usual we write $\langle P, V\rangle(x) \in \{0,1\}$ for the output of V after interaction with P on input $x \in L$ (here, V outputs 1 when P convinced V, otherwise V outputs 0).*

Completeness: *If $(x, w) \in R$, then $\Pr(\langle P(w), V\rangle(x) = 1) = 1$ with $P(w)$ meaning that P has private input w.*

Soundness: *There exists an expected polynomial time knowledge extractor E that, given input $x \in L$ and access to an oracle $P'(x)$ (a possibly malicious prover), outputs $E^{P'(x)}(x) \in W(x) \cup \{\perp\}$ and a real witness with probability*

$$\Pr(E^{P'(x)}(x) \in W(x)) \geq \Pr(\langle P', V\rangle(x) = 1).$$

Here, the extraction of a witness $w \in W(x)$ through E for a given $x \in L$ from a prover P represents the *knowledge* of the machine P. However, Cramer [CDS94] introduced a generalization of this concept which we call Σ-protocols.

Definition 19. *A* **Σ-protocol** *is a 3-move proof of knowledge in the common reference string (CRS) model. Given an element x and some* NP*-language L, a prover P who knows a witness for $x \in L$ (i. e. he has a tuple $(x, w) \in R$) wants to convince a verifier V that $x \in L$ (i. e. that $w \in W(x)$). The protocol works as follows.*

- *Both, V and P have access to a CRS $\sigma \in \{0,1\}^*$.*
- *P sends an initial message a to V.*
- *V sends a random challenge e to P.*
- *P sends an answer z to V.*
- *The proof p then is $p = (a, e, z)$.*

V then evaluates (σ, x, p) and decides whether to accept or reject. Thereby, the protocol has the three following properties.

Completeness: *If* $(x, w) \in R$, *then* $\Pr(\langle P(w), V\rangle(x) = 1) = 1$ *with* $P(w)$ *meaning that* P *has private input* w.

Special Soundness: *There exists a PPT knowledge extractor* E *that, given two acceptable transcripts of conversations* (a, e, z) *and* (a, e', z'), *where* $e \neq e'$, *between any prover* P' *and* V, *can compute a witness* $w \in W(x)$.

Special Honest Verifier Zero-Knowledge: *There exists a PPT simulator* S *that, given* x *and* e, *can create a triple* (a, e, z) *which is indistinguishable from a real proof with challenge* e.

A concrete realization of a Σ-protocol is the following protocol of SCHNORR [CDS94, Section 2.1] which proofs knowledge of a discrete logarithm in a cyclic group $G = \langle g\rangle$ of prime order q, $g \neq 1_G$.

Example 5. *Let* $G = \langle g\rangle$ *be a cyclic group of prime order* q *and* $g \neq 1_G$. *Then the following protocol* π_{SCHNORR} *is a* Σ*-protocol.*

Input: $L := G$ *and* $x := g^w \in L$, *where* $w \in \{0, \ldots, q-1\}$ *is given as private input to* P, *then:*

- P *chooses* $\ell \in_R \{2, \ldots, q-1\}$ *and sends* $a := g^\ell$ *to* V;
- V *sends* $e \in_R \{2, \ldots, q-1\}$ *to* P;
- P *sends* $z = (\ell + e \cdot w) \bmod q$ *to* V;
  V *checks if* $g^z = a \cdot x^e$;

Figure 2: Protocol π_{SCHNORR} (Schnorr's Σ-protocol)

Proof. Completeness is satisfied since

$$g^z = g^{(\ell + e\cdot w) \bmod q} = g^{(\ell + e\cdot w)} = g^\ell \cdot (g^w)^e = a \cdot x^e.$$

Special soundness is also immediate since if we have two transcripts (a, e, z) and (a, e', z') this implies

$$z = \ell + e \cdot w \bmod q \text{ and } z' = \ell + e' \cdot w \bmod q, \tag{3.2}$$

which yields $w = (z - z') \cdot (e - e')^{-1} \bmod q$. For the special honest verifier zero-knowledge property consider the simulator S which, given access to x and e generates the triple (a, e, z) with $a := g^z x^{-e}$ by choosing $z \in_R \{0, \ldots, q-1\}$ at random. This proof is acceptable since

$$a \cdot x^e = g^z x^{-e} \cdot x^e = g^z$$

and indistinguishable from a real proof due to the fact that z can be chosen freely as so does an honest V in the protocol. □

With the aid of a random oracle $\mathcal{O}$ we can transform an arbitrary Σ-protocol into a non-interactive zero-knowledge proof in the sense of the following theorem.

Theorem 2. *Given a Σ-protocol π and a random oracle $\mathcal{O}$, it is possible to make π non-interactive. To achieve this, P gets the challenge e by querying $\mathcal{O}$ with (x, a, aux), where aux is some auxiliary public string. Then, P computes z and the proof is set to $p = (a, z, aux)$. Finally, V can verify p by querying $\mathcal{O}$ with (x, a, aux) and then using π's verifier. The resulting non-interactive proof system then satisfies the following three properties:*

Completeness: *If $(x, w) \in R$, then $\Pr(\langle P(w), V\rangle(x) = 1) = 1$ with $P(w)$ meaning that P has private input w.*

Soundness: *For any $x \notin L$ and any prover P' it holds that $\Pr(\langle P', V\rangle(x) = 1) \leq \frac{1}{3}$.*

Zero-Knowledge: *There exists a PPT simulator $S^{\mathcal{O}}$ that, given $x \in L$ can create a triple (a, z, aux) that is indistinguishable from a real proof, provided it can modify the oracle such that it may give (x, a, aux, e) to $\mathcal{O}$ and if (x, a, aux) has not been queried before, $\mathcal{O}$ answers with e.*

Proof. For the completeness property it is nothing to show.

To prove soundness let $x \notin L$ and P' be any prover. Since $x \notin L$ there exists no witness $w \in W(x)$ for x as $W(x) = \emptyset$. Hence, to every commitment a of P' there exists at most one valid challenge e since otherwise we may, by running the protocol again with a, extract two different challenges $e \neq e'$ and thus compute a witness by the special soundness property of π. Consequently

$$\Pr(\langle P', V\rangle(x) = 1) \leq \frac{1}{|\operatorname{im}\mathcal{O}|},$$

where $\operatorname{im}\mathcal{O}$ is the image of $\mathcal{O}$ since P' must decide for a challenge e which is correct with probability at most $\frac{1}{|\operatorname{im}\mathcal{O}|}$.

We finally prove the zero-knowledge property of the system. For this purpose we define the required simulator $S^{\mathcal{O}}$ interacting with a possibly malicious verifier V'. Concretely, the simulator $S^{\mathcal{O}}$ gets as input the values σ, x and aux and proceeds as follows.

1. $S^{\mathcal{O}}$ chooses a challenge e randomly.
2. $S^{\mathcal{O}}$ calculates a proof (a, e, z) for $x \in L$ by invoking the simulator S^{π} provided by the special honest verifier zero-knowledge property of π.
3. $S^{\mathcal{O}}$ gives (x, a, aux, e) to the oracle $\mathcal{O}$.
4. $S^{\mathcal{O}}$ outputs the proof (a, z, aux).

The simulator $S^{\mathcal{O}}$ is clearly PPT as so is S^{π} and outputs a convincing proof (a, z, aux) indistinguishable from a real proof of V' interacting with a real prover P. The indistinguishability follows from the special honest verifier zero-knowledge property of π and the fact that it is impossible for V' to query the oracle $\mathcal{O}$ with (x, a, aux) beforehand, since a cheating V' cannot guess a in advance, except with negligible probability.

□

Since the UC security model explained in Section 3.1 is simulation based, we need to provide a simulator $\mathcal{S}$ in the ideal world that fares as well as any adversary $\mathcal{A}$ in the real world. In particular, if we make use of non-interactive zero-knowledge proofs based on Σ-protocols in the sense of Theorem 2, a simulator needs to extract witnesses for simulated proofs made by an adversary on behalf of corrupt voters. This motivates the need of some type of witness extended emulation for a proof as in Theorem 2 similar to those in proofs of knowledge (Definition 18) or Σ-protocols. Concretely we have to be aware that there exists a witness extractor that outputs proofs p for elements $x \in L$, indistinguishable from real proofs *together with witnesses* w for memberships of $x \in L$. This will finally yield non-malleability of our proof system as desired. As GROTH has proved [Gro04, Theorem 1] this is possible for the non-interactive zero-knowledge proofs we use (i. e. as defined within Theorem 2) and for completeness we want to present the theorem and its proof here.

Before stating the theorem we fix some notation: Given some algorithm A we denote by $A^{\mathcal{O}}$ the modification where A has access to a random oracle $\mathcal{O}$. Then, $y \leftarrow A^{\mathcal{O}}(z)$ denotes the probability distribution on the output y depending on the input z. Finally,

$$\Pr(y \leftarrow A^{\mathcal{O}}(z) : \ C(y,z)) \tag{3.3}$$

denotes the probability that condition $C(y,z) \in \{0,1\}$ – depending on y and z – is satisfied after the random experiment $y \leftarrow A^{\mathcal{O}}(z)$. If we consider inputs z which are polynomial in $k \in \mathbb{N}$, i. e. $z \in \bigcup_{\kappa \leq k^d} \{0,1\}^\kappa$ for some $d \in \mathbb{N}$, we can regard (3.3) as a function in $k \in \mathbb{N}$ with values in $[0,1]$ by abusing notation and thus denoting with (3.3) the probability $\Pr(z \in_R \bigcup_{\kappa \leq k^d} \{0,1\}^\kappa ; \ y \leftarrow A^{\mathcal{O}}(z) : \ C(y,z))$ for fixed $k \in \mathbb{N}$. We then write

$$\Pr(y \leftarrow A^{\mathcal{O}}(z) : \ C_A(y,z)) \approx \Pr(y \leftarrow B^{\mathcal{O}}(z) : \ C_B(y,z))$$

if and only if $|\Pr(y \leftarrow A^{\mathcal{O}}(z) : \ C_A(y,z)) - \Pr(y \leftarrow B^{\mathcal{O}}(z) : \ C_B(y,z))|$ is negligible in $k \in \mathbb{N}$ (c. f. Remark 2).

Theorem 3. *Let A be an adversary that generates valid Σ-protocol proofs $p = (p_1, \ldots, p_m)$ for elements $x = (x_1, \ldots, x_m) \in L^m$. Then, for every such adversary there exists an expected polynomial time emulator E_A such that for all (even unbounded) distinguisher machines $\mathcal{D}$ it holds*

$$\begin{aligned} &\Pr((x,p,s) \leftarrow A^{\mathcal{O}}(z) : \ (x,p) \in V \wedge \mathcal{D}^{\mathcal{O}}(x,p,z,s) = 1) \\ &\qquad \approx \ \Pr((x,p,w,s) \leftarrow E_A^{\mathcal{O}}(z) : \ (x,p) \in V \wedge (x,w) \in W \wedge \mathcal{D}^{\mathcal{O}}(x,p,z,s) = 1). \end{aligned} \tag{3.4}$$

Here, z is polynomial in k, $\mathcal{O}$ is a random oracle, V is the set of vector pairs (x,p) such that p_i is a valid proof for $x_i \in L$, W is the set of pairs (x,w), where $w = (w_1, \ldots, w_m)$ and w_i is a witness for $x_i \in L$ and $s \in \{0,1\}^*$ is some auxiliary output.

Proof. To this end, we assume w. l. o. g. that A *always* queries $\mathcal{O}$ to make a valid proof instead of simply guessing the oracle answer. As guessing has the negligible probability $\frac{1}{|\operatorname{im} \mathcal{O}|}$ we do not restrict generality. Let z be bounded in k by some polynomial $P(k)$.

Input : z of length $|z| = P(k)$
Output: (x, p, w) with $(x, p) \in V$ and $(x, w) \in W$

1 **run** $A^{\mathcal{O}}(z)$ to get $(x = (x_1, \ldots, x_m), p = (p_1, \ldots, p_m))$ and while running, save the states $\mathcal{S}_1, \ldots, \mathcal{S}_m$ of A whenever it queries $\mathcal{O}$
2 **for** $i = 1$ **to** m **do**
3 *// try to find a witness for the proof p_i made with q_i*
4 $(a, z, aux) := p_i$
5 $(x_i, a, aux) := q_i$
6 $e := \mathcal{O}(q_i)$
7 **run** a simulation of $A^{\mathcal{O}}(z)$ in state $\mathcal{S}_i$, now giving random answers e' to subsequent queries to $\mathcal{O}$, and get (x, p')
8 **if** A uses $e' \neq e$ to produce the valid proof p_i' **then**
9 extract witness w_i by using proofs p_i and p_i' together with the special soundness property
10 **else**
11 **goto** line 7;
12 **end**
13 **end**
14 $w := (w_1, \ldots, w_m)$

Figure 3: Algorithm E_A

The algorithm E_A clearly produces valid proofs p for the vector x indistinguishable from those proofs made by A for any distinguisher $\mathcal{D}$. Moreover, E_A can extract witnesses $w_1, \ldots, w_m$ with overwhelming probability due to the fact that we can choose $e' \neq e$ by random with overwhelming probability $\frac{|\operatorname{im}\mathcal{O}|-1}{|\operatorname{im}\mathcal{O}|}$. It remains to be shown that E_A runs in expected polynomial time.

To prove that, let $t_A(k)$ be the running time of A on input z with $|z| = P(k)$, $t_{E_A}(k)$ be the running time of E_A and $t^7_{E_A}(k)$ be the running time of line 7. Furthermore, let $\rho \in (0, 1]$ be the probability that A uses a challenge to generate a valid proof after querying the oracle. The random variable X that describes the experiment when A uses a challenge is geometrically distributed and thus E_A has to run A in line 7 expected $\mathbb{E}[X] = 1/\rho$ times to get an oracle answer that is used by A to generate a valid proof. Hence

$$\mathbb{E}[t^7_{E_A}] \leq \rho \cdot \mathbb{E}[X] \cdot t_A(k) = t_A(k) \tag{3.5}$$

is an upper bound for the expected running time in line 7. Finally, A can obviously query the oracle at most $t_A(k)$ times which implies $m \leq t_A(k)$ and thus the **for**-loop could be run at most $t_A(k)$ times, which finally yields

$$\mathbb{E}[t_{E_A}] = m \cdot \mathbb{E}[t^7_{E_A}] + t_A(k) + m \cdot t_E \overset{(3.5)}{\leq} m \cdot (t_A(k) + t_E) + t_A(k) \leq t_A(k)^2 + t_A(k)(t_E + 1),$$

where t_E is the running time of the polynomial time knowledge extractor E provided by the special soundness property of the Σ-protocol. The last equation finally completes the proof. □

3.2.3. The Ideal Voting Functionality

We finally present the ideal voting functionality $\mathcal{F}_{\text{VOTING}}$ from [Gro04, p. 8]. As we will see in Section 3.3 this functionality has a UC-realization through a hybrid protocol $\pi^{\mathcal{F}_{\text{KM}}}_{\text{VOTING}}$. We stress that the following functionality is not perfect as it ignores a series of security properties described in Section 2.1 such as incoercibility and universal verifiability which are crucial for e-voting protocols. Nevertheless it is not trivial to prove that such a minimal ideal voting functionality is UC-realizable and thus it is a desirable first approach. However, possible extensions and discussions on how to realize incoercibility are postponed to the last chapter of this thesis.

Definition 20. *The ideal voting functionality $\mathcal{F}_{\text{VOTING}}$ is defined as follows.*

Parties *: adversary $\mathcal{S}$, $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities)*

- *Upon receiving* (**vote**, *sid*, (V_i,*v*)) *from V_i store it and send* (**vote**, *sid*, V_i) *to $\mathcal{S}$. Ignore subsequent* **vote** *messages from V_i with* SID *sid.*
- *Upon receiving* (**no-block**, *sid*, V_i) *from $\mathcal{S}$, check if some* (**vote**, *sid*, (V_i,*v*)) *has been stored. In that case add v to the result σ. Ignore subsequent* **no-block** *messages for V_i with* SID *sid from $\mathcal{S}$.*
- *Upon receiving* (**result**, *sid*) *from $\mathcal{S}$ compute the final result σ_{FINAL}, store it and send* (**result**, *sid*, σ_{FINAL}) *to $\mathcal{S}$.*
- *Upon receiving* (**deliver-result**, *sid*, A_j) *from $\mathcal{S}$, send* (**result**, *sid*, σ_{FINAL}) *to A_j.*

Figure 4: Functionality $\mathcal{F}_{\text{VOTING}}$

3.3. A Voting Protocol in the UC Framework

Within this section we finally present the protocol that UC-realizes the ideal voting functionality $\mathcal{F}_{\text{VOTING}}$ presented in Section 3.2. The presented proofs and gists were extracted from [Gro04]. However, the purpose of this section is to expound intricacies within the presented protocol and proofs in [Gro04] along with a small simplification of the constructed simulator.

3.3.1. A Message Board Functionality

We want to concentrate on the main voting functionalities within the voting protocol and thus make use of the universal composition Theorem 1. Namely, we move some functionalities required by the protocol into an ideal functionality $\mathcal{F}_{\text{KM}}$ that represents a fictitious message board where every party participating within the voting protocol can post *authenticated* messages but cannot delete them. Moreover, it supplies us with an ideal key distribution for an arbitrary homomorphic threshold cryptosystem as in Definition 17.

Let us first present a definition of $\mathcal{F}_{\mathrm{KM}}$. A discussion of the presented functionality then follows.

Definition 21. *The ideal message board functionality $\mathcal{F}_{\mathrm{KM}}$ is defined as follows.*

Parties *: adversary $\mathcal{A}$, $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities)*

- *Generate keys $(pk, vk_1, \ldots, vk_n, sk_1, \ldots, sk_n)$ for the homomorphic threshold cryptosystem as in Definition 17.*
  Send (**public key**, sid, pk) *to* $V_1, \ldots, V_m, A_1, \ldots, A_n$ *and* $\mathcal{A}$.
  Send (**verification keys**, $sid, (vk_1, \ldots, vk_n)$) *to* $A_1, \ldots, A_n$ *and* $\mathcal{A}$.
  Send (**secret share**, sid, sk_i) *to* A_i, $i = 1, \ldots, n$.
- *Upon receiving* (**message**, sid, m) *from* V_i *store* (**message**, $sid, (V_i, m)$) *and send it to* $\mathcal{A}$.
- *Upon receiving* (**no-block**, $sid, (V_i, m)$) *from* $\mathcal{A}$, *check if some* (**message**, $sid, (V_i, m)$) *has been stored. In that case store* (**post**, $sid, (V_i, m)$). *Ignore subsequent* **no-block** *messages for* V_i *with* SID *sid from* $\mathcal{A}$.
- *Upon receiving* (**tally**, sid) *from* $\mathcal{A}$ *send all stored* (**post**, $sid, (V_i, m)$) *messages to* $A_1, \ldots, A_n$. *Ignore subsequent* (**tally**, sid) *messages.*
- *Upon receiving* (**post**, sid, m) *from* A_i *send* (**post**, $sid, (A_i, m)$) *to* $A_1, \ldots, A_n$ *and* $\mathcal{A}$.

Figure 5: Functionality $\mathcal{F}_{\mathrm{KM}}$

The ideal functionality $\mathcal{F}_{\mathrm{KM}}$ is an important building block within the construction of the e-voting hybrid protocol $\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}$ in the next section. In particular, it already includes crucial design patterns, namely we can subsume the following observations.

Remark 3. *Due to Definition 21 the following observations hold.*

1. **Authenticated messages:** *We assume every message posted on the message board to be authenticated by simply requiring that the identity of the sender is broadcast together with the message. Within $\mathcal{F}_{\mathrm{KM}}$, this is implicitly done by specifying from which party the message was send. In the UC framework this is modeled via a direct channel between the party and the functionality which is feasible due to the definitions of the hybrid model and an ideal functionality (c. f. Definition 13 and Definition 10, respectively). In practice this will be usually done by using digital signatures.*
2. **Blocking votes:** *We allow the adversary to block votes. This allows us to model real-life intricacies such as unstable internet connections or faulty computers of voters which affect the availability property of the overall voting system. However, the message board does not support blocking of authorities. As* Groth *[Gro04, p. 7] argues, this is reasonable since a voting system has to ensure that all authorities have appropriate back-up procedures to ensure that they all could establish the necessary communication.*

3. **UC-realizability of $\mathcal{F}_{\mathrm{KM}}$:** *As we use $\mathcal{F}_{\mathrm{KM}}$ as a building block in the hybrid voting protocol being constructed within the next section, it is reasonable to ask whether and how this ideal functionality is UC-realizable. A discussion on this issue is given in Section 3.4.*

3.3.2. The Voting Protocol

Assume the existence of a special encoding function $\mathrm{Enc} : (\mathcal{V}, +) \longrightarrow (\mathcal{M}, +)$ used to encode the vote space $\mathcal{V}$. For a discussion on such an encoding function see Remark 4. We are now in position to present the e-voting protocol $\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}$ which is a $\mathcal{F}_{\mathrm{KM}}$-hybrid protocol in the sense of Definition 13.

Definition 22. *Assume we have given a Σ-protocol π_Σ, a homomorphic (t, n)-threshold cryptosystem (E, D) and a set of possible votes $\mathcal{V}$. The $\mathcal{F}_{\mathrm{KM}}$-hybrid protocol $\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}$ is then defined as follows.*

Parties *: $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities)*
Access to*: $\mathcal{F}_{\mathrm{KM}}$ and a random oracle $\mathcal{O}$*

(V1) Invoke $\mathcal{F}_{\mathrm{KM}}$ to establish the key generation and distribution.
As a result, each party obtains the public key pk and each authority obtains the verification keys and its secret key.

(V2) Each V_i with pk on its `IC` *tape and a valid vote v_i on the* `input` *tape computes $x_i := E(pk, \mathrm{Enc}(v_i))$, then chooses some a_i at random and queries $\mathcal{O}$ with (x_i, a_i, aux_i) to get a challenge e_i, where $aux_i := (pk, sid, V_i)$.*
Finally, V_i creates a proof $p_i = (a_i, z_i, aux_i)$ that $v_i \in \mathcal{V}$ using π_Σ.

V_i then sends $(\mathbf{message}, sid, (x_i, p_i))$ to $\mathcal{F}_{\mathrm{KM}}$.

(V3) When authority A_j has pk and the verification key vk_j on the `IC` *tape it does the following. When receiving a bunch of votes it extracts those votes $(v_{i_1}, \ldots, v_{i_\ell})$ with valid proofs. It then computes $C := \prod_{k=1}^{\ell} v_{i_k}$ and the corresponding decryption share ds_j. Finally, he chooses some a_j at random, queries $\mathcal{O}$ with (ds_j, a_j, aux_j) to get a challenge e_j, where $aux_j := (pk, sid, A_j)$ and creates a proof p_j using π_Σ and its verification key vk_j.*

A_j then sends $(\mathbf{post}, sid, (ds_j, p_i))$ to $\mathcal{F}_{\mathrm{KM}}$.

(V4) Each authority A_j uses the first t decryption shares $ds_{j_1}, \ldots, ds_{j_t}$ of C with valid proofs to decrypt C to $\mathrm{Enc}(\sigma) := D(pk, C)$.

A_j then outputs $(\mathbf{result}, sid, \sigma)$.

Figure 6: Protocol $\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}$

We make the following remark concerning the encoding function Enc in the protocol.

Remark 4. *We discuss how to encode the vote space $\mathcal{V}$ from which an eligible voter can choose its vote. Since the concrete definition of such an encoding function* Enc *depends profoundly on the specific cryptosystem it is necessary to determine some important properties of* Enc *though. Namely, given the encryption function $E : \mathcal{K} \times (\mathcal{M}, +) \longrightarrow (\mathcal{C}, \cdot)$ of the cryptosystem,* Enc *has to fulfill the following properties, assuming by simplicity that $\mathcal{V}$ holds an additive group structure:*

1. Enc *must efficiently encode every possible vote $v \in \mathcal{V}$ by a value* $\mathrm{Enc}(v) \in \mathcal{M}$*, i. e.* Enc *has to be an efficient function* $\mathrm{Enc} : (\mathcal{V}, +) \longrightarrow (\mathcal{M}, +)$.
2. Enc *must be a group homomorphism.*
3. *The preimage of an* Enc*-encoded result has to be efficiently extractable.*
4. Enc *should not derogate the security of the cryptosystem.*

However, such an encoding function exists. An example can be found in [Gro04, pp. 8].

The successive section is now devoted to the security proof for $\pi^{\mathcal{F}_{\mathrm{KM}}}_{\mathrm{VOTING}}$, i. e. we will show that $\pi^{\mathcal{F}_{\mathrm{KM}}}_{\mathrm{VOTING}}$ UC-realizes $\mathcal{F}_{\mathrm{VOTING}}$ in the $\mathcal{F}_{\mathrm{KM}}$-hybrid model.

3.3.3. The Security Proof

As already mentioned, the constructed $\mathcal{F}_{\mathrm{KM}}$-hybrid e-voting protocol $\pi^{\mathcal{F}_{\mathrm{KM}}}_{\mathrm{VOTING}}$ is secure in the sense that it UC-realizes the ideal functionality $\mathcal{F}_{\mathrm{VOTING}}$ in the case where less than t authorities were corrupted and the adversaries are non-adaptive. This is the gist of the following main theorem [Gro04, Theorem 2].

Theorem 4. *The $\mathcal{F}_{\mathrm{KM}}$-hybrid e-voting protocol $\pi^{\mathcal{F}_{\mathrm{KM}}}_{\mathrm{VOTING}}$ UC-realizes $\mathcal{F}_{\mathrm{VOTING}}$ for the class of non-adaptive adversaries that corrupt less than t authorities.*

Proof. By Definition 12 we have to provide an ideal process adversary $\mathcal{S}$ for every non-adaptive adversary $\mathcal{A}$ and every environment $\mathcal{Z}$ such that the distribution ensembles $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ and $\mathrm{EXEC}_{\mathrm{IDEAL}_{\mathcal{F}},\mathcal{S},\mathcal{Z}}$ are indistinguishable (c. f. Definition 8). Of course, we can restrict us to a special class of adversaries. Consider an adversary $\mathcal{D}$ which simply forwards every message given on its `IC` tape to $\mathcal{Z}$ and forwards every input given on its `input` tape to the specified party. Such an adversary is called *dummy adversary*, c. f. [Can01, p. 45], and the clue is that $\mathcal{D}$ fares as well as any adversary:

Lemma 1. [Can01, Claim 10] *Let π and ρ be PPT protocols. Then π UC-emulates ρ according to Definition 9 if and only if there exists a PPT adversary $\mathcal{S}$ such that*

$$\mathrm{EXEC}_{\pi,\mathcal{D},\mathcal{Z}} \approx \mathrm{EXEC}_{\rho,\mathcal{S},\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

As a result we do only have to provide a simulator $\mathcal{S}$ for the dummy adversary $\mathcal{D}$. More particular we may identify $\mathcal{D}$ and $\mathcal{Z}$ in the following and describe a simulator $\mathcal{S}$ operating in the ideal process that fares as well as $\mathcal{D}$ in the real world in the sense that it is infeasible for $\mathcal{Z}$ to decide whether it interacts with $\mathcal{D}$ or $\mathcal{S}$.

$\mathcal{S}$ operates in the ideal process with dummy voters $\widetilde{V_1}, \ldots, \widetilde{V_m}$ and dummy authorities $\widetilde{A_1}, \ldots, \widetilde{A_n}$. Further:

- $\mathcal{S}$ controls the oracle $\mathcal{O}$ (i. e. $\mathcal{S}$ can assign a response to any query). In particular, $\mathcal{S}$ thus can simulate π_Σ-proofs.
- $\mathcal{S}$ runs a simulated $\mathcal{F}_{\text{KM}}$-hybrid execution of $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$. The parties within this simulation are denoted with $V_1, \ldots, V_m$ and $A_1, \ldots, A_n$. Thereby, $\mathcal{S}$ forwards every message between $\mathcal{D}$ and $\mathcal{Z}$.
- In particular, $\mathcal{S}$ simulates the invocation of $\mathcal{F}_{\text{KM}}$. As a result, $\mathcal{S}$ knows the secret shares of the private key sk of all authorities. Hence $\mathcal{S}$ can now decrypt messages encrypted under the public key.

Additionally, $\mathcal{S}$ has to handle five specific events.

$\mathcal{S}1$: *Corruption.*
If voter V_i is corrupted, i. e. $\mathcal{Z}$ sends $(\textbf{message}, sid, (x_i, p_i))$ and $(\textbf{no-block}, sid, (V_i, x_i, p_i))$ to $\mathcal{F}_{\text{KM}}$ on behalf of V_i, then $\mathcal{S}$ checks if p_i is a valid proof. In that case, $\mathcal{S}$ decrypts x_i to get V_i's vote v_i. Then, $\mathcal{S}$ delivers $(\textbf{vote}, sid, (V_i, v_i))$ and $(\textbf{no-block}, sid, V_i)$ to $\mathcal{F}_{\text{VOTING}}$ on behalf of $\widetilde{V_i}$.

$\mathcal{S}2$: *Honest votes.*
Upon receiving $(\textbf{vote}, sid, V_i)$ from $\mathcal{F}_{\text{VOTING}}$, $\mathcal{S}$ knows that $\widetilde{V_i}$ got $(\textbf{vote}, sid, (V_i, v_i))$ as input from $\mathcal{Z}$ but it does *not* know the vote v_i.
If V_i has not yet received the public key, $\mathcal{S}$ ignores this case.
Otherwise $\mathcal{S}$ must simulate V_i trying to cast a vote. Hence it forms $x_i := E(pk, v_0)$ for an arbitrary $v_0 \in \mathcal{V}$ and simulates a proof p_i for x_i representing a valid vote.
$\mathcal{S}$ then passes $(\textbf{message}, sid, (V_i, x_i, v_i))$ as input to $\mathcal{F}_{\text{KM}}$ and sends it to $\mathcal{Z}$.
If $\mathcal{S}$ receives $(\textbf{no-block}, sid, (V_i, m))$ from $\mathcal{Z}$ afterwards, $\mathcal{S}$ sends it to $\mathcal{F}_{\text{KM}}$ and $(\textbf{no-block}, sid, V_i)$ to $\mathcal{F}_{\text{VOTING}}$.

$\mathcal{S}3$: *Correct tallying (1).*
Upon $\mathcal{Z}$ sending $(\textbf{tally}, sid)$ to $\mathcal{F}_{\text{KM}}$, $\mathcal{S}$ lets $\mathcal{F}_{\text{KM}}$ send the stored messages $(\textbf{post}, sid, (V_i, x_i, p_i))$ to $A_1, \ldots, A_n$.
It sends $(\textbf{result}, sid)$ to $\mathcal{F}_{\text{VOTING}}$ and learns σ_{FINAL}.
Let $C := \prod_{k=1}^{\ell} x_{i_k}$ be the product of all votes with valid proofs. $\mathcal{S}$ uses the simulation property of the threshold cryptosystem to simulate shares ds_j and proofs of correctness p_j for the honest A_j's such that C decrypts to σ_{FINAL}.

$\mathcal{S}4$: *Correct tallying (2).*
After $\mathcal{Z}$ has delivered both, the keys and the messages to honest A_j then $\mathcal{S}$ delivers $(\textbf{post}, sid, (A_j, ds_j, p_j))$ to $\mathcal{F}_{\text{KM}}$ on behalf of A_j.

$\mathcal{S}5$: *Correct tallying (3).*
After A_j has received both, the public key and t decryption shares, $\mathcal{S}$ sends $(\textbf{deliver-result}, sid, A_j)$ to $\mathcal{F}_{\text{VOTING}}$ such that $\widetilde{A_j}$ receives σ_{FINAL}.

Figure 7: Adversary $\mathcal{S}$

The simulator $\mathcal{S}$ defined in Figure 7 has to simulate outputs and messages of participating (possibly corrupted) parties including the transcripts from the ideal functionality $\mathcal{F}_{\text{KM}}$ in the real world model. Thus, we have to prove that any environment $\mathcal{Z}$ cannot distinguish between an execution of the $\mathcal{F}_{\text{KM}}$-hybrid $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ with dummy adversary $\mathcal{D}$ and an execution of $\text{IDEAL}_{\mathcal{F}_{\text{VOTING}},\mathcal{S},\mathcal{Z}}$ with simulator $\mathcal{S}$. To prove this, we start with the $\mathcal{F}_{\text{KM}}$-hybrid execution and inductively construct experiments EXP^i (where EXP^0 is the initial $\mathcal{F}_{\text{KM}}$-hybrid experiment) indistinguishable for any environment $\mathcal{Z}$ from EXP^{i-1} leading to an experiment $\text{EXP}^4 \approx \text{IDEAL}_{\mathcal{F}_{\text{VOTING}},\mathcal{S},\mathcal{Z}}$. Within this process we use expected polynomial time algorithms and this however is feasible as long as these algorithms are not used within the ideal process itself. We finally obtain

$$\text{EXEC}_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{D},\mathcal{Z}} \approx \text{EXP}^1 \approx \text{EXP}^2 \approx \text{EXP}^3 \approx \text{EXP}^4 \approx \text{IDEAL}_{\mathcal{F}_{\text{VOTING}},\mathcal{S},\mathcal{Z}},$$

which completes the proof.

First note that $\mathcal{Z}$ has only knowledge about at most $t-1$ secret shares so that it is infeasible for $\mathcal{Z}$ to gain any information about encrypted votes. Moreover, there exist at least t honest authorities and w. l. o. g. we may enumerate them by $A_1, \ldots, A_\ell$ where $\ell \geq t$. Moreover let $V_1, \ldots, V_{m_1}$ and $V_{m_1+1}, \ldots, V_m$, where $m_1 + m_2 = m$, denote the corrupted and uncorrupted voters, respectively.

Experiment EXP^1

In EXP^1 we want to modify the $\mathcal{F}_{\text{KM}}$-hybrid model such that we know the result of the election without using messages posted on the message board.

Consider the situation after $\mathcal{Z}$ has submitted $(\textbf{tally}, sid)$ to $\mathcal{F}_{\text{KM}}$. We denote with $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ the first-step simulator depending on $\mathcal{D}$ (and consequently on $\mathcal{Z}$) provided with the ability to control the random oracle $\mathcal{O}$. $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ proceeds as follows: it calculates the result σ_{FINAL} of the election by extracting the votes $v_{m_1+1}, \ldots, v_m$ at the `input` tapes of the honest voters being not blocked and adds them to the votes $v_1, \ldots, v_{m_1}$ of corrupted voters. To get these votes it has to decrypt them using the secret shares $sk_1, \ldots, sk_\ell$ of the honest authorities since corrupted votes $E(pk, \text{Enc}(v_i))$, $i = 1, \ldots, v_{m_1}$, where casted by $\mathcal{Z}$ (or $\mathcal{D}$ accordingly) and thus $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ cannot simply read-off them. Therefore, $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ obtains

$$\sigma_{\text{FINAL}} = \sum_{i=1}^{m_1} v_i + \sum_{j=m_1+1}^{m} v_j$$

as the result of the election and it now has to simulate decryption shares $ds'_1, \ldots, ds'_\ell$ for the honest authorities since otherwise $\mathcal{Z}$ could distinguish between the encrypted result $c'_{\sigma_{\text{FINAL}}} := E(pk, \text{Enc}(\sigma_{\text{FINAL}}))$ by $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ and the encrypted result $c_{\sigma_{\text{FINAL}}}$ given to it in the real-life, i. e. when interacting with $\mathcal{D}$ instead of $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$. To simulate the decryption shares, $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ uses the simulator S_E provided by the simulation property of the threshold cryptosystem (c. f. Definition 16). It gives S_E the values $c'_{\sigma_{\text{FINAL}}}$ and σ_{FINAL} together with the decryption shares $ds_1, \ldots, ds_{n-\ell}$ of the corrupted authorities as input and obtains shares $ds'_1, \ldots, ds'_\ell$

which decrypt the encrypted sum $c'_{\sigma_{\text{FINAL}}}$ of the votes $v_1, \ldots, v_m$ to σ_{FINAL}. Using the ability to control the random oracle $\mathcal{O}$ it additionally simulates proofs p'_j for the decryption shares being correct. Finally, $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ writes the new tuples (ds'_j, p'_j), $j = 1, \ldots, \ell$, onto the `input` tapes of the honest authorities. Then, the experiment EXP^1 continues as the real-life experiment. Now, we claim that

$$\text{EXEC}_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{D},\mathcal{Z}} \approx \text{EXP}^1_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{S}_1^{\mathcal{D},\mathcal{O}},\mathcal{Z}},$$

where $\text{EXP}^1_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{S}_1^{\mathcal{D},\mathcal{O}}}$ denotes the output of $\mathcal{Z}$ after the experiment EXP^1 described above. To prove that we consider the values where $\mathcal{Z}$ might distinguish, possibly depending on the security parameter $k \in \mathbb{N}$ and the input $z = (v_1, \ldots, v_m)$ given to $\mathcal{Z}$. These are:

- The new decryption shares $ds'_1, \ldots, ds'_\ell$ and the corresponding proofs $p'_1, \ldots, p'_\ell$ given as `input` to the honest authorities.
- The controlled oracle $\mathcal{O}$ in the case where $\mathcal{Z}$ knows a value a used inside π_Σ beforehand.

First, it is clear that the outcome σ_{FINAL} of the new experiment is the same as in the initial experiment. Let now D_k^1 and D_k^2 be distinguishing machines for decryption shares of the cryptosystem and π_Σ-proofs, respectively, possibly depending on the security parameter $k \in \mathbb{N}$. Then

$$\begin{aligned} \mu_1(k) &:= |\Pr(D_k^1(z, ds) = 1) - \Pr(D_k^1(z, ds') = 1)| \text{ and} \\ \mu_2(k) &:= |\Pr(D_k^2(z, p) = 1) - \Pr(D_k^2(z, p') = 1)| \end{aligned}$$

are negligible for all $z \in \mathcal{V}^m$ by the simulation property of the threshold cryptosystem and the zero-knowledge property of the π_Σ-proofs, respectively, where ds' and p' are the vectors of the simulated proofs while ds and p are the shares and proofs used within the original experiment.

Finally the random value a used within the π_Σ-proofs cannot be guessed by $\mathcal{Z}$ beforehand since it is possible to choose a from a superpolynomial space $\mathcal{P}$. Hence $\mathcal{Z}$ cannot detect that $\mathcal{O}$ is controlled, except with negligible probability $1/|\mathcal{P}|$. Thus, we obtain

$$|\text{EXEC}_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{D},\mathcal{Z}}(k, z) - \text{EXP}^1_{\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}},\mathcal{S}_1^{\mathcal{D},\mathcal{O}},\mathcal{Z}}(k, z)| < \mu_1(k) + \mu_2(k) + 1/|\mathcal{P}|$$

for sufficiently large $k \in \mathbb{N}$ and $z \in \mathcal{V}^m$ as desired.

Experiment EXP^2

In EXP^2 we want to emulate votes of corrupted voters using witness extraction.

More specifically, $\mathcal{Z}$ may corrupt voters $V_1, \ldots, V_{m_1}$ and cast votes on behalf of them (e. g. this is the case if each corrupted voter V_i is coerced to send its entire state to $\mathcal{D}$). Hence, it must be possible for a simulator to *extract* the votes $v'_1, \ldots, v'_{m_1}$ casted by $\mathcal{D}$ on behalf of corrupt voters. In order to prove this, we extend the simulator $\mathcal{S}_1^{\mathcal{D},\mathcal{O}}$ to a second-step simulator $\mathcal{S}_2^{\mathcal{D},\mathcal{O}}$, now modifying the execution process of EXP^1.

Let A be the algorithm which gets as input the configuration $\mu_{\mathcal{Z}}$ of $\mathcal{Z}$ and a set $\mathcal{M}$ of (**message**, sid, (x_i, p_i)) messages and then runs the entire execution within the interval between key generation and $\mathcal{Z}$ not yet having submitted (**tally**, sid). Finally, A outputs $\mu'_{\mathcal{Z}}$ which now contains the encrypted votes $x' := (x'_1, \ldots, x'_{m_1})$ and proofs $p' := (p'_1, \ldots, p'_{m_1})$ of the corrupted voters.

Due to Theorem 3, we can replace A by an expected polynomial time machine E_A that produces output (x'', p'', w) consisting of the encrypted corrupted votes x'', their proofs p'' made by $\mathcal{Z}$ and the *witnesses* w which contain the votes of the corrupt parties, indistinguishable from the output of A.

Finally, $\mathcal{S}_2^{\mathcal{D},\mathcal{O}}$ proceeds as follows: after key generation has been done, $\mathcal{S}_2^{\mathcal{D},\mathcal{O}}$ pre-generates for each honest voter and each possible vote it can get the (**message**, sid, (x_i, p_i)) messages; let $\mathcal{M}$ denote this set of messages. It then reads off $\mu_{\mathcal{Z}}$ and gives this together with $\mathcal{M}$ as input to E_A. The algorithm E_A then outputs the tuple (x'', p'', w) and a configuration $\mu'_{\mathcal{Z}}$ of $\mathcal{Z}$ after the interval. $\mathcal{S}_2^{\mathcal{D},\mathcal{O}}$ then restarts $\mathcal{Z}$ with configuration $\mu'_{\mathcal{Z}}$ and uses these votes w to calculate the outcome as in EXP^1 but without the need to decrypt votes using the honest authorities' secret shares of the private key. We do now claim, that

$$\mathrm{EXP}^1_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_1^{\mathcal{D},\mathcal{O}}, \mathcal{Z}} \approx \mathrm{EXP}^2_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_2^{\mathcal{D},\mathcal{O}}, \mathcal{Z}}\,. \tag{3.6}$$

In fact, w is a vector containing the votes corresponding to the corrupt voters. This is true because the used π_Σ-proofs are by definition proofs of knowledge for the relation

$$R := \bigcup_{v \in \mathcal{V}} \{(E(pk, \mathrm{Enc}(v)), v)\}$$

The indistinguishability of $\mathrm{EXP}^1_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_1^{\mathcal{D},\mathcal{O}}, \mathcal{Z}}$ and $\mathrm{EXP}^2_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_2^{\mathcal{D},\mathcal{O}}, \mathcal{Z}}$ is finally a direct consequence of Theorem 3. In fact, the only point at which $\mathcal{Z}$ could distinguish is the difference of the outputs of A and E_A and hence for all $d \in \mathbb{N}$, $k \in \mathbb{N}$ and $z \in \bigcup_{\kappa \leq k^d} \{0,1\}^\kappa$ it holds that

$$|\,\mathrm{EXP}^1_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_1^{\mathcal{D},\mathcal{O}}, \mathcal{Z}}(k, z) - \mathrm{EXP}^2_{\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}, \mathcal{S}_2^{\mathcal{D},\mathcal{O}}, \mathcal{Z}}(k, z)| \leq \rho_d(k),$$

where $\rho_d(k) \in [0,1]$ denotes the probability

$$\begin{aligned}\rho_d(k) := |\,&\Pr((x, p, \mu'_{\mathcal{Z}}) \leftarrow A^{\mathcal{O}}(Z) :\ (x,p) \in V \wedge \mathcal{D}^{\mathcal{O}}(x, p, \mu'_{\mathcal{Z}}, Z) = 1) \\ &- \Pr((x, p, w, \mu'_{\mathcal{Z}}) \leftarrow E_A^{\mathcal{O}}(Z) :\ (x,p) \in V \wedge (x,w) \in W \wedge \mathcal{D}^{\mathcal{O}}(x, p, \mu'_{\mathcal{Z}}, Z) = 1)|,\end{aligned} \tag{3.7}$$

where $Z := (\mu_{\mathcal{Z}}, \mathcal{M})$ is such that $Z \in \bigcup_{\kappa \leq k^d} \{0,1\}^\kappa$, c. f. the remarks following equation (3.3) on page 30. Now, ρ_d is negligible for every $d \in \mathbb{N}$ by Theorem 3 and this completes the proof of (3.6).

Experiment EXP^3

In EXP^3 we cast default votes instead of letting honest voters cast their real vote known to the environment $\mathcal{Z}$ but unknown to the simulator.

Concretely, we select a $v_0 \in \mathcal{V}$ at random and extend the simulator from experiment EXP^3 to the third-step simulator $\mathcal{S}_3^{\mathcal{D},\mathcal{O}}$ which proceeds as follows. Instead of the $(\mathbf{message}, sid, (x_i, p_i))$ messages subsumed within $\mathcal{M}$, $\mathcal{S}_3^{\mathcal{D},\mathcal{O}}$ calculates a set $\mathcal{M}'$ which contains for every honest voter V_i, $i = m_1 + 1, \ldots, m$, the encrypted v_0-vote $x_i' := E(pk, \mathrm{Enc}(v_0))$ together with a simulated π_Σ-proof p_i' of correctness. The simulator gives $\mathcal{M}'$ instead of $\mathcal{M}$ to A and then proceeds as $\mathcal{S}_2^{\mathcal{D},\mathcal{O}}$ does. This construction corresponds to the fact, that we need to produce votes that look like real votes in the ideal process given by the ideal functionality $\mathcal{F}_{\mathrm{VOTING}}$. From this construction, we obtain

$$\mathrm{EXP}^2_{\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}, \mathcal{S}_2^{\mathcal{D},\mathcal{O}}, \mathcal{Z}} \approx \mathrm{EXP}^3_{\pi_{\mathrm{VOTING}}^{\mathcal{F}_{\mathrm{KM}}}, \mathcal{S}_3^{\mathcal{D},\mathcal{O}}, \mathcal{Z}} \tag{3.8}$$

by the following argument. First, the outcome σ_{FINAL} in this experiment is the same as before since we still use the real votes to produce the result at the end of the experiment and thus the outcome is the same as in the experiments before. What has changed is that $\mathcal{S}_3^{\mathcal{D},\mathcal{O}}$ casts v_0-votes instead of the real votes $v_{m_1+1}, \ldots, v_m$. However, since the used public-key cryptosystem is semantically secure it holds that

$$|\Pr(D_k^E(\mathcal{M}) = 1) - \Pr(D_k^E(\mathcal{M}') = 1)|$$

is negligible in $k \in \mathbb{N}$ for every PPT distinguishing machine D_k^E depending on the security parameter $k \in \mathbb{N}$, where the probability is taken over all possible encryption keys pk with length k, c. f. Definition 16. This clearly implies (3.8).

Experiment EXP^4

In EXP^4 we can finally establish the ideal process by going back to use the honest authorities' secret shares to decrypt the votes of corrupt voters with valid proofs.

Consider the simulator $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$ that acts like $\mathcal{S}_3^{\mathcal{D},\mathcal{O}}$, except that it, instead of invoking the witness simulator E_A, decrypts votes with valid proofs sent to him by $\mathcal{D}$ on behalf of corrupt voters $V_1, \ldots, V_{m_1}$ using the secret shares $sk_1, \ldots, sk_\ell$ of the honest authorities. Now, roughly speaking, $\mathcal{Z}$ cannot distinguish, whether the simulation of the interval after key generation having been done and before the $(\mathbf{tally}, sid)$ message was sent to $\mathcal{F}_{\mathrm{KM}}$, has been done by E_A or A (Theorem 3) neither if it has been done by A or by really simulating this interval. More specific, the difference in the outputs of E_A and A again reduces to the probability given in (3.7) while the configuration $\mu_{\mathcal{Z}}$ of $\mathcal{Z}$ after the execution of the interval and $\mu_{\mathcal{Z}}'$ calculated by A are actually the same.

Therefore, $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$ can switch back to simply decrypting votes and simulating the execution in the interval without $\mathcal{Z}$ being able to tell the difference to the experiment EXP^3. Finally, we can change this modification of EXP^3 so as to yield $\mathrm{IDEAL}_{\mathcal{F}_{\mathrm{VOTING}}, \mathcal{S}, \mathcal{Z}}$. We simply consider the ideal functionality $\mathcal{F}_{\mathrm{VOTING}}$ together with the dummy parties $\widetilde{V_1}, \ldots, \widetilde{V_m}$ and $\widetilde{A_1}, \ldots, \widetilde{A_n}$. Then

- let $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$ send a v_0-vote as in EXP^3 if an uncorrupted $\widetilde{V_i}$ voted (as in $\mathcal{S}2$),
- let $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$ decrypt a corrupted and unblocked vote with valid proof and send it to $\mathcal{F}_{\mathrm{VOTING}}$ (as in $\mathcal{S}1$),

- when $\mathcal{Z}$ submitted $(\textbf{tally}, sid)$ to $\mathcal{F}_{\text{KM}}$ we act as in EXP^1 and finally submit the outcome σ_{Final} to $\mathcal{F}_{\text{Voting}}$ (this subsumes the actions done by $\mathcal{S}$ within $\mathcal{S}3$-$\mathcal{S}5$).

The experiment EXP^4 with simulator $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$ and environment $\mathcal{Z}$ and the ideal process $\text{IDEAL}_{\mathcal{F}_{\text{Voting}},\mathcal{S},\mathcal{Z}}$ with simulator $\mathcal{S}$ and environment $\mathcal{Z}$ are then finally the same. The only difference is the way how the result σ_{Final} is obtained by $\mathcal{S}$ or $\mathcal{S}_4^{\mathcal{D},\mathcal{O}}$, respectively, but the results are however actually the same. This finally completes the proof of the Theorem. □

Let us finally give a brief discussion on the e-voting protocol $\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}$. To our knowledge, the protocol is indeed the first protocol that UC-realizes a voting functionality but it however leaves a series of problems open. We first stress that, on the one hand, it is easy to handle *adaptive corruption of voters* by simply requiring that they erase their plaintext vote and randomness after encrypting their vote. This was also proposed in [Gro04, p. 14] since it seems to be reasonable as assuming immediate erasing implies a "rudimentary receipt-freeness". On the other hand it is not trivial to handle *adaptive corruption of authorities* as the example in [Gro04, p. 13] illustrates. Dealing with this problem is a little more difficult and Groth proposes two approaches. The first is to guarantee security against adaptive corruption of authorities by evaluating in the erasure model, i. e. where an adversary does *not* learn the entire history of a corrupted party but its current state and tape contents. Finally, the second deals with the case of the erasure-free model where the adversary learns the entire history of a corrupted party using Pallier encryption [Gro04, p. 14]. Although the argumentations are a little sketchy it seems to be possible to achieve security against adaptive corruption of authorities.

The protocol $\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}$ guarantees important security properties such as good availability, secrecy, correctness and authenticity among others (such as accuracy, fairness and robustness) due to the fact that these properties are already provided by the UC framework and the fact, that the ideal voting functionality $\mathcal{F}_{\text{Voting}}$ is defined to achieve these requirements. However, there are still some issues open: even if the protocol $\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}$ guarantees verifiability due to the fact that the message board is readable by everyone, it does not guarantee universal verifiability in the sense that everyone can verify the outcome of the election. Next, incoercibility does not play a role in any of the considerations and it is unclear whether it is possible to modify the protocol such that it becomes impossible for an adversary to coerce a voter.

3.4. A UC-realization of the Message Board

This section provides a discussion on how to UC-realize the ideal functionality $\mathcal{F}_{\text{KM}}$ in Definition 21. As we have seen within the previous sections, the functionality $\mathcal{F}_{\text{KM}}$ can be used to construct a voting protocol $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ that UC-realizes $\mathcal{F}_{\text{VOTING}}$ in the $\mathcal{F}_{\text{KM}}$-hybrid model and it is thus reasonable to ask, whether this functionality is UC-realizable, i. e. if there exists a protocol π_{KM} which UC-realizes $\mathcal{F}_{\text{KM}}$ with respect to Definition 12. Once we have proven this, the protocol $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ can be extended to a full-blown electronic voting protocol by the universal composition theorem (Theorem 1).

The main intention of this section is to prove the following insights: As it turns out, such a protocol π_{KM} exists, if we use a threshold cryptosystem that uses a discrete logarithm as shared secret key such as the ELGAMAL cryptosystem, c. f. Example 2 in Section 2.2.2. In the case, where a cryptosystem uses a RSA-modulus, i. e. a product $N = pq$ for large primes p and q, as public key, such as the PALLIER cryptosystem, c. f. Example 3, it is not clear, if such a protocol exists.

Let us briefly explain the reasons of the results just stated, followed by a detailed analysis of the problem and the proofs for the assertions presented within the former paragraph. The idea is to split the message functionality $\mathcal{F}_{\text{KM}}$ into two functionalities $\mathcal{F}_{\text{DKG}}$ and $\mathcal{F}_{\text{BB}}$, where the first is the distributed key generation (DKG) functionality which is clearly contained within $\mathcal{F}_{\text{KM}}$ and $\mathcal{F}_{\text{BB}}$ is a bare message board used to publicly broadcast messages from and to parties in the network. Details of this approach are discussed later, but we stress here that we need to UC-realize the DKG functionality in order to obtain a full protocol π_{KM} which UC-realizes $\mathcal{F}_{\text{KM}}$ and this is the problematic step. Namely, the protocols which may realize $\mathcal{F}_{\text{DKG}}$ depend on the underlying cryptosystem, since they need to provide the corresponding keys of that system and this is the crucial point. Namely, considering cryptosystems which use discrete logarithms as shared secret keys – such as ELGAMAL – it turns out that there indeed exist DKG protocols which UC-realize $\mathcal{F}_{\text{DKG}}$, c. f. [Wik05, AF04]. For the readers convenience and in order to include our notations we present the protocol of WIKSTRÖM in [Wik05] since, in the original paper, it contains some misleading index errors[2]. Unfortunately, the ELGAMAL system is only useful in the setting of electronic elections if we use the version of Example 2, i. e. where a multiplication of cryptotexts yields an encryption of the product of the plaintexts, but then, tallying cannot be carried out as explained in Section 3.3. This drawback is explained in detail within Section 3.4.3.1. However, most voting protocols in the literature use factorization as a trapdoor for the keys to be shared, and it is thus necessary to discuss UC-secure DKG for these systems as well. However, the situation is much more difficult in this case and unclear, if common distributed key generation protocols for that class of cryptosystems such as [Sho00, DK01, BF97, FS01, FMY98] UC-realize an ideal DKG-functionality $\mathcal{F}_{\text{DKG}}$. A discussion on this problem can be found in Section 3.4.3.2.

This section is structured as follows: We explain our idea to realize the message board

[2] Concretely, wrong indices where included in steps 8 and 16 of the original protocol and no explicit presentation of the sets I_1 and I_2 was given.

functionality $\mathcal{F}_{\text{KM}}$ in Section 3.4.1. Namely, we split the functionality into several, less extensive functionalities and then prove that these are UC-realizable within the Sections 3.4.2 and 3.4.3. However, Section 3.4.3 additionally provides a discussion on the problem explained above, i. e. the dependence of the UC-realizability of $\mathcal{F}_{\text{DKG}}$ on the underlying cryptosystem. Finally, the main intention of this section is to obtain a full-blown electronic voting protocol π_{VOTING} that UC-realizes $\mathcal{F}_{\text{VOTING}}$ without the use of ideal functionalities other than a common reference string (CRS). We point out that we need to slightly modify the proposed functionalities and protocols by Groth, presented in Section 3.3, in order to obtain the aspired UC-realizability results.

3.4.1. Preliminaries

As a first step, we add an additional party T, the *tallying supervisor*, to the functionality $\mathcal{F}_{\text{KM}}$. The party T does nothing else but sending the $(\textbf{tally}, sid)$ message to $\mathcal{F}_{\text{KM}}$ when the election ends. Then $\mathcal{F}_{\text{KM}}$ forwards this message to the adversary and waits for the $(\textbf{no-block}, sid, \textbf{tally})$ message from it. This adaption is necessary to obtain control over the finalization of the election within the security proofs. Moreover, this party is also added to the ideal voting functionality $\mathcal{F}_{\text{VOTING}}$, but now sending $(\textbf{result}, sid)$.

Since a message delivery cannot be secured in the UC-framework as the adversary is always allowed to block messages from parties in the real world execution, we need to restrict the adversary to block at most $(n-t)$ authorities statically. To this end, we write $\mathbb{A}$ for the set of possibly blocked authorities, i. e. we have $\mathbb{A} \subset \{A_1, \ldots, A_n\}$ and $|\mathbb{A}| \leq n-t$. As a consequence, $\mathcal{F}_{\text{KM}}$ has to be modified such that the $(\textbf{post}, sid, (A_i, m))$ messages in $\mathcal{F}_{\text{KM}}$ are not sent to all $A_1, \ldots, A_n$, but to those authorities not included $\mathbb{A}$. The modified functionalities can be found within the Appendix A.1.

We now decompose $\mathcal{F}_{\text{KM}}$ (c. f. Definition 21) in an obvious way into three ideal functionalities $\mathcal{F}_{\text{DKG}}$, $\mathcal{F}_{\text{BB}}$ and $\mathcal{F}_{\text{BC}}$, i. e. we obtain

$$\mathcal{F}_{\text{KM}} = \mathcal{F}_{\text{DKG}} + \mathcal{F}_{\text{BB}} + \mathcal{F}_{\text{BC}}. \tag{3.9}$$

As a consequence, we need to slightly modify $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ and this can be found in the Appendix A.1. Here, the functionality $\mathcal{F}_{\text{DKG}}$ guarantees a distributed key generation and is defined as follows.

Definition 23. *The distributed key generation functionality $\mathcal{F}_{\text{DKG}}$ is defined as follows.*

Parties *: adversary $\mathcal{S}$, $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities)*

- *Generate keys $(pk, vk_1, \ldots, vk_n, sk_1, \ldots, sk_n)$ for the homomorphic threshold cryptosystem as in Definition 17.*
  Send $(\textbf{public key}, sid, pk)$ to $V_1, \ldots, V_m, A_1, \ldots, A_n$ and $\mathcal{S}$.
  Send $(\textbf{verification keys}, sid, (vk_1, \ldots, vk_n))$ to $A_1, \ldots, A_n$ and $\mathcal{S}$.
  Send $(\textbf{secret share}, sid, sk_i)$ to A_i, $i = 1, \ldots, n$.
- *Then halt.*

Figure 8: Functionality $\mathcal{F}_{\text{DKG}}$

The existence of protocols realizing this functionality is proven in Section 3.4.3. The second functionality $\mathcal{F}_{\text{BB}}$ implements a bulletin board, which receives messages from voters and, if not blocked by the adversary using a *delayed output*[3], delivers them to the authorities at the end of the election.

Definition 24. *The bulletin board functionality $\mathcal{F}_{\text{BB}}$ is defined as follows.*

Parties *: adversary $\mathcal{S}$, $V_1, \ldots, V_m$ (voters), $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor)*

- *Upon receiving* $(\textbf{message}, sid, m)$ *from* V_i*, store* $(\textbf{message}, sid, (V_i, m))$ *and send it to* $\mathcal{S}$*.*
- *Upon receiving* $(\textbf{no-block}, sid, (V_i, m))$ *from* $\mathcal{S}$*, check if* $(\textbf{message}, sid, (V_i, m))$ *is stored. In that case, store* $(\textbf{post}, sid, (V_i, m))$ *and ignore future* $(\textbf{no-block}, sid, (V_i, m))$ *messages from* $\mathcal{S}$*.*
- *Upon receiving* $(\textbf{tally}, sid)$ *from* T*, send it to* $\mathcal{S}$*. Ignore subsequent* $(\textbf{tally}, sid)$ *requests.*
- *Upon receiving* $(\textbf{no-block}, sid, \textbf{tally})$ *from* $\mathcal{S}$*, write each* $(\textbf{post}, sid, (V_i, m))$ *message to the* `SO` *tapes of the authorities as delayed output.*

Figure 9: Functionality $\mathcal{F}_{\text{BB}}$

The existence of protocols realizing this functionality is proven in Section 3.4.2. In order to present a protocol π_{BB} that UC-realizes the bulletin board functionality $\mathcal{F}_{\text{BB}}$, we need the following broadcast functionality $\mathcal{F}_{\text{BC}}$ which was first investigated in [GL05, pp. 266] and [CLOS02, pp. 502] that secures an authenticated broadcast channel.

Definition 25. *The broadcast functionality $\mathcal{F}_{\text{BC}}$ is defined as follows.*

Parties *: adversary $\mathcal{S}$, $V_1, \ldots, V_m$ (voters), $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor). Set $P_i := V_i$, $i = 1, \ldots, m$, and $P_{m+1} := T$.*

- *Upon receiving* $(\textbf{broadcast}, sid, (\mathcal{P}, m))$ *from party* P_i *for* $\mathcal{P} \subseteq \{A_1, \ldots, A_n, P_1, \ldots, P_{m+1}\}$*, check if* m *is of the form* $m = (\textbf{msg-string}, \widetilde{sid}, \widetilde{m})$*. In that case, store* $(\textbf{broadcast}, sid, (P_i, \mathcal{P}, m))$ *and send it to* $\mathcal{S}$*.*
- *Upon receiving* $(\textbf{no-block}, sid, (P_i, \mathcal{P}', m))$ *from* $\mathcal{S}$*, check if a broadcast message* $(\textbf{broadcast}, sid, (P_i, \mathcal{P}, m))$ *with* $\mathcal{P}' \subseteq \mathcal{P}$ *has been stored. In that case, send* $m = (\textbf{msg-string}, \widetilde{sid}, (P_i, \widetilde{m}))$ *to all parties in* $\mathcal{P}'$*. Halt.*

Figure 10: Functionality $\mathcal{F}_{\text{BC}}$

[3] When a functionality $\mathcal{F}$ sends a *delayed output* m to a set $\mathcal{P} = \{P_1, \ldots, P_k\}$ of parties, this is defined as $\mathcal{F}$ sending $(\mathcal{P}, m)$ to the adversary $\mathcal{A}$. Whenever $\mathcal{A}$ answers with $(\textbf{no-block}, (P_i, m))$, the message m is delivered to P_i, i. e. written on its `SO` tape, c. f. [Can01, p. 68].

Since it is up to the adversary $\mathcal{S}$ to deliver messages, the only purpose of this functionality is to guarantee that no two parties, being not blocked by the adversary, receive two different messages with the same SID *sid*. We stress that a single instance of $\mathcal{F}_{\mathrm{BC}}$ can only be invoked once, i. e. for each broadcast sent by a party, a new instance of $\mathcal{F}_{\mathrm{BC}}$ has to be invoked. We remind the reader that this is done by the dummy parties, which invoke the ideal functionality $\mathcal{F}_{\mathrm{BC}}$, c. f. Definition 13 on page 23.

3.4.2. Realizing the Bulletin Board Functionality

We will prove that the bulletin board functionality $\mathcal{F}_{\mathrm{BB}}$ is UC-realizable by a $\mathcal{F}_{\mathrm{BC}}$-hybrid protocol $\pi_{\mathrm{BB}}^{\mathcal{F}_{\mathrm{BC}}}$. Thus we may prove first, that there exists a protocol π_{BC} that UC-realizes the ideal broadcast functionality $\mathcal{F}_{\mathrm{BC}}$.

Definition 26. *The protocol π_{BC} is defined as follows.*

Parties: *$V_1, \ldots, V_m$ (voters), $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor). Set $P_i := V_i$, $i = 1, \ldots, m$, and $P_{m+1} := T$.*

Input : *P_k has $(\textbf{broadcast}, sid, (\mathcal{P}, m))$ on its* `input` *tape.*

1. *P_k sends $x^k := (\mathcal{P}, m)$ to all parties in $\mathcal{P}$.*
2. *Party P_i with $i \in \{1, \ldots, m+1\} \setminus \{k\}$ does nothing.*
3. *Upon A_j receiving a value $x^j = (\mathcal{P}^j, m^j)$ from P_k, it checks if $A_j \in \mathcal{P}^j$ and if m^j is of the form $m^j = (\textbf{msg-string}, \widetilde{sid}, \widetilde{m})$. In that case, it sends x^j to all parties in $\mathcal{P}^j$.*
4. *Party A_j waits for a message $x^{j,\ell}$ from every $A_\ell \in \mathcal{P}^j \setminus \{V_k\}$. When received $x^{j,1}, \ldots, x^{j,k-1}, x^{j,k+1}, \ldots, x^{j,n}$, then A_j outputs (P_k, m^j) if and only if $x^j = x^{j,1} = \ldots = x^{j,k-1} = x^{j,k+1} = \ldots = x^{j,n}$, otherwise it halts.*

Figure 11: Protocol π_{BC}

The following lemma based on [GL05, Proposition 3.2] asserts now the required facts.

Lemma 2. *The protocol π_{BC} UC-realizes $\mathcal{F}_{\mathrm{BC}}$.*

Proof. The presented protocol π_{BC} and the functionality $\mathcal{F}_{\mathrm{BC}}$ are very similar to those defined in [GL05, pp. 266]. For our purposes, we only added the following refinements which are necessary in our security proofs.

Refinement 1: In $\mathcal{F}_{\mathrm{BC}}$ we allow a party P_k to send a message to a subset $\mathcal{P}$ of parties and not necessarily to all parties, c. f. [CLOS02, p. 502].

Refinement 2: We explicitly give the adversary $\mathcal{S}$ the possibility to block messages. In [GL05] however this is assumed implicitly since in older versions of the UC-framework it was possible for the adversary to hold back messages of parties which is not possible in the new version [Can01] that we presented in Definition 11 of

IDEAL$_{\mathcal{F}}$. Nevertheless, as explained in [Can01, p. 51, Footnote 13], this gap can be filled by using *delayed outputs*.

Refinement 3: In order to simplify the embedding of $\mathcal{F}_{\text{BC}}$ into our protocols, we assume that a broadcast message in turn contains a message of the form (**msg-string**, $\widetilde{sid}, \widetilde{m}$) which is finally sent to all parties.

The above-mentioned refinements clearly do not affect the security proof in [GL05, pp. 267]. This is immediate since Refinement 2 only makes $\mathcal{F}_{\text{BC}}$ conform to the newest version [Can01] of the UC-framework and Refinement 3 can be included in the security proof by defining the broadcast message x to be of the required form. Refinement 1 does not affect the security proof of [GL05, pp. 267], since, if a corrupted A_j does not discard the message but send it anyway, this has no effect on the outputs of the parties in the last step of the protocol. Therefore, the proof of [GL05, Proposition 3.2] can be adapted to our slightly modified protocol, which completes our proof. □

Using this lemma, we can now construct a $\mathcal{F}_{\text{BC}}$-hybrid protocol $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$ that UC-realizes $\mathcal{F}_{\text{BB}}$ since we are aware of the fact that $\mathcal{F}_{\text{BC}}$ is UC-realizable.

Definition 27. *The protocol $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$ is defined as follows.*

Parties *: $V_1, \ldots, V_m$ (voters), $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor).*

Access to*: $\mathcal{F}_{\text{BC}}$*

1. *Each V_i with* (**message**, sid, m) *on its* `input` *tape sends* (**broadcast**, sid_{loc_1}, ($\{A_1, \ldots, A_n\}$, (**post**, sid, m))) *to* $\mathcal{F}_{\text{BC}}$.
2. *When T has* (**tally**, sid) *on its* `input` *tape, then T sends* (**broadcast**, sid_{loc_2}, ($\{A_1, \ldots, A_n\}$, (**tally**, sid))) *to* $\mathcal{F}_{\text{BC}}$.
3. *When A_j has* (**tally**, sid) *on its* `SO` *tape, then A_j outputs all received* (**post**, sid, (V_i, m)) *messages.*

Figure 12: Protocol $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$

We can now state the following theorem.

Theorem 5. *The protocol $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$ UC-realizes $\mathcal{F}_{\text{BB}}$ in the $\mathcal{F}_{\text{BC}}$-hybrid model with respect to adversaries that statically block less than $n - t + 1$ authorities.*

Proof. We need to provide an ideal world simulator $\mathcal{S}$, which fares as well as any adversary $\mathcal{A}$ in the real world $\mathcal{F}_{\text{BC}}$-hybrid execution of $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$, i. e. such that

$$\text{EXEC}_{\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{BB}}, \mathcal{S}, \mathcal{Z}}$$

for all environments $\mathcal{Z}$. However, for the proof of security we consider again the dummy adversary $\mathcal{D}$ (c. f. Lemma 1 on page 35) instead of an arbitrary adversary $\mathcal{A}$. Thus, we identify $\mathcal{D}$ and $\mathcal{Z}$ as in the proof of Theorem 4.

Denoting with $\overline{\mathbb{A}} := \{A_1, \ldots, A_n\} \setminus \mathbb{A}$ the complement set, the ideal process adversary $\mathcal{S}$ is defined as follows.

$\mathcal{S}$ operates in the ideal process with dummy voters $\widetilde{V_1}, \ldots, \widetilde{V_m}$, dummy authorities $\widetilde{A_1}, \ldots, \widetilde{A_n}$ and dummy tallying authority $\widetilde{T}$. Further:

- $\mathcal{S}$ runs a simulated $\mathcal{F}_{\text{BC}}$-hybrid execution of $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$ with dummy adversary $\mathcal{D}$. The parties within this simulation are denoted with $V_1, \ldots, V_m$, $A_1, \ldots, A_n$ and T.
- $\mathcal{S}$ forwards every message between $\mathcal{D}$ and $\mathcal{Z}$.

Additionally, $\mathcal{S}$ has to handle two specific events.

$\mathcal{S}1$: *Reaction on a message from $\widetilde{V_i}$.*
Upon receiving $(\textbf{message}, sid, (V_i, m))$ from $\mathcal{F}_{\text{BB}}$, $\widetilde{V_i}$ sent a message.
Then, $\mathcal{S}$ sends $(\textbf{broadcast}, sid_{\text{loc}_1}, (\{A_1, \ldots, A_n\}, (\textbf{post}, sid, m)))$ to $\mathcal{F}_{\text{BC}}$ on behalf of V_i.
When $\mathcal{Z}$ responds with $(\textbf{no-block}, sid_{\text{loc}_1}, (V_i, \overline{\mathbb{A}}, (\textbf{post}, sid, m)))$, $\mathcal{S}$ knows that the message was delivered.
It therefore sends $(\textbf{no-block}, sid, (V_i, m))$ to $\mathcal{F}_{\text{BB}}$ at this point.

$\mathcal{S}2$: *Reaction on a message from $\widetilde{T}$.*
Upon receiving $(\textbf{tally}, sid)$ from $\mathcal{F}_{\text{BB}}$, $\widetilde{T}$ sent the **tally**-message to $\mathcal{F}_{\text{BB}}$.
Then, $\mathcal{S}$ sends $(\textbf{broadcast}, sid_{\text{loc}_2}, (\{A_1, \ldots, A_n\}, (\textbf{tally}, sid)))$ to $\mathcal{F}_{\text{BC}}$ on behalf of T.
When $\mathcal{Z}$ responds with $(\textbf{no-block}, sid_{\text{loc}_2}, (T, \overline{\mathbb{A}}, (\textbf{tally}, sid)))$, then $\mathcal{S}$ knows that the tally ended.
It therefore sends $(\textbf{no-block}, sid, \textbf{tally})$ to $\mathcal{F}_{\text{BB}}$ at this point.

Figure 13: Adversary $\mathcal{S}$

It remains to be shown that

$$\text{EXEC}_{\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}, \mathcal{D}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{BB}}, \mathcal{S}, \mathcal{Z}}$$

which is immediate. Namely, in the $\mathcal{F}_{\text{BC}}$-hybrid execution, $\mathcal{Z}$ may only block messages sent by parties to the ideal functionality $\mathcal{F}_{\text{BC}}$. But whenever $\mathcal{Z}$ blocks a party, the simulator $\mathcal{S}$ behaves exactly as $\mathcal{D}$ does. Thus, the proof of indistinguishability is straightforward and trivial since the delayed messages in $\text{EXEC}_{\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}, \mathcal{D}, \mathcal{Z}}$ are delayed by $\mathcal{S}$, too. □

Remark 5. *Note that the set $\mathbb{A}$ of blocked authorities in the class of adversaries that statically block less than $n-t+1$ authorities is only limited to fulfill $|\mathbb{A}| \leq n-t$ since this is necessary in our setting where the authorities need to decrypt a ciphertext distributedly such that a threshold of t authorities not blocked by the adversary is required. However, Theorem 5 remains true, if we restrict the adversary only to statically block authorities without any assumption on the set of these blocked parties, i. e. it holds:* The protocol $\pi_{\text{BB}}^{\mathcal{F}_{\text{BC}}}$ UC-realizes $\mathcal{F}_{\text{BB}}$ in the $\mathcal{F}_{\text{BC}}$-hybrid model with respect to adversaries that statically block authorities.

3.4.3. Realizing the Distributed Key Generation Functionality

This section is devoted to the discussion on how to realize the distributed key generation functionality $\mathcal{F}_{\text{DKG}}$. As mentioned above, we will explain in the case of cryptosystems that use the discrete logarithm as trapdoor to share the public and the private key of the cryptosystem, it is possible to UC-realize $\mathcal{F}_{\text{DKG}}$ through a protocol π_{DKG}. Unfortunately, most voting protocols choose a different approach, since they use a RSA-modulus $N = pq$ with large primes p and q as public key together with additional keys, whose security depends on the difficulty to factorize N. For such keys, it is not known, if there exist DKG protocols that UC-realize $\mathcal{F}_{\text{DKG}}$.

The first subsection deals with the former class of cryptosystems while the second discusses the class of cryptosystems which use factorizations as trapdoor to share the keys.

3.4.3.1. Cryptosystems using Discrete Logarithms

This section deals with the case, where we assume a public-key cryptosystem (E, D) in which the public key has the form $y = g^x$, for some generator $g \in G$ of a cyclic group G. These cryptosystems are called *discrete logarithm* (DL) based cryptosystems and the most famous example was already defined in Example 2 in Section 2.2.2. Namely, the ELGAMAL cryptosystem uses an element $y = g^x \in G$ of a cyclic group G of prime order q as public key while the encryption works as follows. For a plaintext $m \in G$ the value

$$E_{\text{EG}}(m, r) := (c_1, c_2) := (g^r, my^r), \tag{3.10}$$

with $r \in_R \{2, \ldots, q-1\}$ being some random blinding value, is the corresponding ciphertext. This cryptosystem is useful within mix-nets as it has an appropriate property but, it is well-suited for electronic elections using homomorphic encryption as well. Indeed, the encryption function E_{EG} is homomorphic in the sense that it holds

$$E_{\text{EG}}(m_1, r) \cdot E_{\text{EG}}(m_2, s) = (g^{r+s}, m_1 \cdot m_2 y^{r+s}) = E_{\text{EG}}(m_1 \cdot m_2, r + s)$$

for all $r, s \in \{2, \ldots, q-1\}$ and $m_1, m_2 \in G$, i. e. it is a multiplicative homomorphic encryption function. We point out that, if we modify the encryption function by defining

$$E_{\text{lEG}}(m, r) := (c_1, c_2) := (g^r, g^m y^r),$$

then this results in an additive homomorphic encryption, usually called *lifted* ELGAMAL cryptosystem. Unfortunately, this system is not well-suited for our purposes, since the decryption procedure is too cumbersome as we need to calculate the discrete logarithm of g^m to obtain the plaintext m. This is, why PENG and BAO [PB11] proposed to use the usual ELGAMAL cryptosystem for electronic elections. Instead of interpreting the product of the encrypted votes as result of the outcome as in Section 3.3, they propose to let the set $\mathcal{V}$ of possible candidates consist of prime numbers $p_1, \ldots, p_\ell$ below some bound. As a result, tallying the encrypted votes $x_i := E_{\text{EG}}(v_i, r_i)$, $i = 1, \ldots, m$, then, omitting details,

works as follows. Each authority calculates

$$\prod_{i=1}^{m} x_i = E_{\mathrm{EG}}\left(\prod_{i=1}^{m} v_i, \sum_{i=1}^{m} r_i\right) = E_{\mathrm{EG}}\left(\prod_{j=1}^{\ell} p_j^{c_j}, \sum_{i=1}^{m} r_i\right) =: E_{\mathrm{EG}}(\sigma, r) \quad (3.11)$$

since $v_i = p_{j(i)}$ for some $j(i) \in \{1, \ldots, \ell\}$. Again, with a single decryption, the authorities obtain the result of the election by decomposing σ into its irreducible factors (i. e. their unique prime factors $p_1, \ldots, p_\ell$). This is possible as we assume, that the given group G is at least a unique factorization domain, c. f. the explanations following (3.1) on page 25. Then the final outcome of the election is the number of votes $c_j \leq m$ for each candidate $j \in \{1, \ldots, \ell\}$. In addition to that, Peng and Bao claim that their system is very effective and practically useful. For details on this electronic voting system we refer the reader to the original paper [PB11].

As the previous considerations indicate, it is worthwhile to analyze how public keys in DL based cryptosystems can be shared such that the resulting key distribution protocol UC-realizes $\mathcal{F}_{\mathrm{DKG}}$. Indeed, as mentioned in [PB11], it turns out that sharing a public key in commonly used homomorphic cryptosystems, which use factorizations of large integers as trapdoor, is very cumbersome and ineffective. Examples for such protocols can be found, however, in Section 3.4.3.2. Here, we concentrate on how to share a public value $y = g^x$ as in the ElGamal scheme between n authorities distributedly such that at least t of them have to cooperate in order to decrypt messages. To explain this, we will in a first step present a DKG protocol $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$ based on [Wik05, Protocol 1] to share a key $y = g^x$ that UC-realizes $\mathcal{F}_{\mathrm{DKG}}$, while we adapt this protocol to our setting. In a next step, we introduce a (t, n) threshold version of the above-mentioned ElGamal cryptosystem with encryption function E_{EG} and show how authorities in an election can cooperate to distributedly decrypt the outcome of the election, e. g. calculated as in (3.11).

Before we present the protocol $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$, we need to define two functionalities used within $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$, i. e. at a first glance, the protocol $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$ is again a hybrid protocol having access to the ideal functionalities $\mathcal{F}_{\mathrm{BC}}$, $\mathcal{F}_{\mathrm{CRS}}$ and $\mathcal{F}_{\mathrm{MMT}}$. The functionality $\mathcal{F}_{\mathrm{CRS}}$ is similar to [Can01, p. 78], while we use a concrete distribution D here, i. e. the uniform distribution over G_q defined below. Let $p \in \mathbb{Z}$ be a *safe prime*, i. e. let $p = 2q + 1$ for a prime q be a large prime number, depending on some security parameter $k \in \mathbb{N}$. Furthermore, let $G_q \subset \mathbb{Z}_p^*$ denote a subgroup of prime order q.

Definition 28. *The common reference string functionality $\mathcal{F}_{\mathrm{CRS}}$ is defined as follows.*

Parties *: adversary $\mathcal{S}$ and $A_1, \ldots, A_n$ (authorities)*

- *Upon receiving* (**CRS**, *sid*)*, check if a tuple* $\mathrm{crs} := (g, h_1, h_2, h_3)$ *is recorded. If not, choose $g, h_1, h_2, h_3 \in_R G_q$ at random. Finally, send* (**delayed-CRS**, *sid*, crs) *to $\mathcal{S}$.*
- *Upon receiving* (**no-block**, *sid*, A_j) *from $\mathcal{S}$, send* (**CRS**, *sid*, crs) *to A_j.*

Figure 14: Functionality $\mathcal{F}_{\mathrm{CRS}}$

The choice of G_q is not arbitrary as it is necessary to choose $p = 2q+1$ carefully such that the *Decisional Diffie-Hellman (DDH) assumption* here below holds in G_q. This turns out to be crucial for the upcoming security proofs, i. e. we need that the following assumption holds, c. f. [Wik05, p. 265].

Definition 29 (Decisional Diffie-Hellman)**.** *Let $e_1, e_2, e_3 \in \mathbb{Z}_q$ be randomly chosen. The* **(non-uniform) Decisional Diffie-Hellman assumption for** *G_q states that for all PPT non-uniform Turing machines A and arbitrary $c > 0$, there exists an $n_0 \in \mathbb{N}$ such that*

$$|\Pr(A(g^{e_1}, g^{e_2}, g^{e_3}) = 1) - \Pr(A(g^{e_1}, g^{e_2}, g^{e_1 e_2}) = 1)| < \frac{1}{n^c}$$

for all $n > n_0$.

Finally, we need the multiple message transmission functionality $\mathcal{F}_{\text{MMT}}$ [Wik05, p. 266] which enables parties in the UC-framework to communicate secretly with each other.

Definition 30. *The multiple message transmission functionality $\mathcal{F}_{\text{MMT}}$ is defined as follows.*

Parties*: adversary $\mathcal{S}$ and $P_1, \ldots, P_\ell$*

- *In the first activation expect to receive a value* $(\textbf{receiver}, sid)$ *from some party P_ν. Then send* $(\textbf{receiver}, sid, P_\nu)$ *to $P_1, \ldots, P_\ell$ and the adversary $\mathcal{S}$. Ignore subsequent* $(\textbf{receiver}, sid)$ *messages.*
- *Upon receiving* $(\textbf{send}, sid, m)$ *from party P_j, send* $(\textbf{send}, sid, (P_j, m))$ *to P_ν and* $(\textbf{send}, sid, (P_j, |m|))$ *to $\mathcal{S}$.*

Figure 15: Functionality $\mathcal{F}_{\text{MMT}}$

For the UC-realizability of $\mathcal{F}_{\text{MMT}}$, we refer to [Wik05, Lemma 2], i. e.

Lemma 3. *There exists a protocol π_{MMT} that UC-realizes $\mathcal{F}_{\text{MMT}}$ under the DDH assumption in G_q.*

We are now in the position to present the protocol $\pi^{\text{DL}}_{\text{DKG}}$. For the sake of simplicity, we write that "*A sends a broadcast m to parties $P_1, \ldots, P_\ell$*" for the fact that A sends a message $(\textbf{broadcast}, sid_{\text{loc}}, (\{P_1, \ldots, P_\ell\}, m))$ to the ideal broadcasting functionality $\mathcal{F}_{\text{BC}}$.

Definition 31. *The protocol $\pi^{\text{DL}}_{\text{DKG}}$ is defined as follows. Let $\{\Omega_1, \Omega_2, \Omega_3\}$ be a partition of $\{1, \ldots, n\}$ such that $||\Omega_a| - |\Omega_b|| \leq 1$ for all $a, b \in \{1, 2, 3\}$.*

Parties *: $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities)*
Access to*: $\mathcal{F}_{\text{BC}}$, $\mathcal{F}_{\text{CRS}}$ and $\mathcal{F}_{\text{MMT}}$*

Authorities: *Each A_j, $j = 1, \ldots, n$, proceeds as in the following steps 1 - 17.*

1. *Send* $(\textbf{receiver}, sid_{\text{loc}_j})$ *to $\mathcal{F}_{\text{MMT}}$.*
2. *Wait for* $(\textbf{receiver}, sid_{\text{loc}_\ell}, A_\ell)$ *for all $\ell \neq j$ from $\mathcal{F}_{\text{MMT}}$.*
3. *Send* $(\textbf{CRS}, sid)$ *to $\mathcal{F}_{\text{CRS}}$ and wait for* $(\textbf{CRS}, sid, (g, h_1, h_2, h_3))$.

Key generation.

4. *If $j \in \Omega_\nu$, set $f(j) := \nu \in \{1,2,3\}$. Choose $a_{j,\tau} \in_R \mathbb{Z}_q$ for $\tau = 0,\dots,t-1$ and define a polynomial $a \in \mathbb{Z}_q[z]$ with $\deg(a) = t-1$ by*

$$a_j(z) := \sum\nolimits_{\tau=0}^{t-1} a_{j,\tau} z^\tau.$$

Further, for all $\ell = 1,\dots,n$:
Set $\alpha_{j,\tau} := h_{f(j)}^{a_{j,\tau}}$ and $s_{j,\ell} := a_j(\ell)$.
Send $(\textbf{send}, sid_{\mathrm{loc}_j}, (\textbf{share}, A_\ell, s_{j,\ell}))$ to $\mathcal{F}_{\mathrm{MMT}}$.
Send broadcast $(\textbf{public-elements}, sid, \{\alpha_{j,\tau}\}_{\tau=0}^{t-1})$ to $A_1,\dots,A_n$.

5. *Upon receiving $(\textbf{public-elements}, sid, (A_\ell, \{\alpha_{j,\tau}\}_{\tau=0}^{t-1}))$ from $\mathcal{F}_{\mathrm{BC}}$ and $(\textbf{send}, sid_{\mathrm{loc}_\ell}, (\textbf{share}, A_\ell, s_{\ell,j}))$ from $\mathcal{F}_{\mathrm{MMT}}$ for $\ell = 1,\dots,n$, verify that $h_{f(\ell)}^{s_{\ell,j}} = \prod_{\tau=0}^{t-1} \alpha_{\ell,\tau}^{j^\tau}$. If so, send broadcast $(\textbf{complaints}, sid, \emptyset)$ to $A_1,\dots,A_n$. Otherwise go to 11.*

6. *Upon receiving $(\textbf{complaints}, sid, (A_\ell, \Delta_\ell))$ from $\mathcal{F}_{\mathrm{BC}}$ (c.f. step 11) for all $\ell \neq j$, check if each $\Delta_\ell = \emptyset$. If so, set $I_1 := \{1,\dots,n\}$, otherwise go to 12.*

7. *Define $s_j := \sum_{\ell \in I_1} s_{\ell,j}$ and $\beta_j := g^{s_j}$.*
Send broadcast $(\textbf{construct-pk}, sid, \beta_j)$ to $A_1,\dots,A_n$.

8. *Upon receiving $(\textbf{construct-pk}, sid, (A_\ell, \beta_\ell))$ for $\ell \in I_1$, verify*

$$\beta_\ell = \prod\nolimits_{i=1}^{t} \beta_i^{\prod_{a \neq i} \frac{\ell - a}{i - a}}$$

for all $\ell \in I_1$. If so, set $I_2 := \{1,\dots,t\}$, otherwise go to 14.

9. *Define*

$$y := \prod_{i \in I_2} \beta_i^{\prod_{\ell \neq i} \frac{\ell}{\ell - i}}$$

and choose

$$vk_j \in_R \mathbb{Z}_q.$$

Send broadcast $(\textbf{public key}, sid, y)$ to $V_1,\dots,V_m$.
Send broadcast $(\textbf{verification key}, sid, vk_j)$ to $V_1,\dots,V_m$.

10. *Upon receiving $(\textbf{verification key}, sid, (A_\ell, vk_\ell))$ from $\mathcal{F}_{\mathrm{BC}}$, $\ell \neq j$, output $(\textbf{public key}, sid, y)$, $(\textbf{verification keys}, sid, (vk_1,\dots,vk_n))$ and $(\textbf{secret share}, sid, s_j)$.*

Handle cheating with the $s_{\ell,j}$ and $\alpha_{j,\tau}$.

11. *Define*

$$\Delta_j := \left\{ \ell \in \{1,\dots,n\} \;\middle|\; h_{f(\ell)}^{s_{\ell,j}} \neq \prod\nolimits_{\tau=0}^{t-1} \alpha_{\ell,\tau}^{j^\tau} \right\}.$$

Send broadcast $(\textbf{complaints}, sid, \Delta_j)$ to $A_1,\dots,A_n$.

12. *Upon receiving* $(\textbf{complaints}, sid, (A_\ell, \Delta_\ell))$ *for* $\ell \neq j$, *define* $\Gamma_j := \{\ell \mid j \in \Delta_\ell\}$.
Send broadcast $(\textbf{refutes}, sid, \{s_{j,\ell}\}_{\ell\in\Gamma_j})$ *to* $A_1, \ldots, A_n$.

13. *Upon receiving* $(\textbf{refutes}, sid, (A_\ell, \{s_{\ell,i}\}_{i\in\Gamma_\ell}))$ *for* $\ell \neq j$, *replace the old* $s_{\ell,j}$ *received in step 4 if* $j \in \Gamma_\ell$. *Then define*

$$I_1 := \left\{\ell \in \{1,\ldots,n\} \mid h_{f(\ell)}^{s_{\ell,i}} = \prod_{\tau=0}^{t-1} \alpha_{\ell,\tau}^{i^\tau} \ \forall i \in \{j\} \cup \bigcup_{k=1}^{n} \Gamma_k\right\}$$

and go to 7.

Handle cheating with the β_ℓ.

14. *Choose* $b_j \in_R \mathbb{Z}_q[z]$ *with* $\deg(b_j) = t-1$ *and* $b_j(0) = s_j$. *Let* $b_j(z) = \sum_{\tau=0}^{t-1} b_{j,\tau} z^\tau$ *and define* $\gamma_{j,\tau} := g^{b_{j,\tau}}$, $\delta_{j,\tau} := h_{f(j)}^{b_{j,\tau}}$ *and* $\zeta_{j,\ell} := b_j(\ell)$.
Send $(\textbf{send}, sid_{\mathrm{loc}_j}, (\textbf{share-2}, A_\ell, \zeta_{j,\ell}))$ *to* $\mathcal{F}_{\mathrm{MMT}}$.
Send broadcast $(\textbf{public-elements-2}, sid, \{\gamma_{j,\tau}, \delta_{j,\tau}\}_{\tau=1}^{t-1})$ *to* $A_1, \ldots, A_n$.

15. *Upon receiving* $(\textbf{public-elements-2}, sid, (A_\ell, \{\gamma_{\ell,\tau}, \delta_{\ell,\tau}\}_{\tau=1}^{t-1}))$ *from* $\mathcal{F}_{\mathrm{BC}}$ *for all* $\ell \in I_1$, *set* $\gamma_{j,0} := \beta_\ell$ *and* $\delta_{j,0} := \prod_{i\in I_1} \alpha_{i,\ell}$. *Then define*

$$\Delta'_j := \left\{\ell \in I_1 \mid g^{\zeta_{\ell,j}} \neq \prod_{\tau=0}^{t-1} \gamma_{\ell,\tau}^{j^\tau} \ \vee \ h_{f(\ell)}^{\zeta_{\ell,j}} \neq \prod_{\tau=0}^{t-1} \delta_{\ell,\tau}^{j^\tau}\right\}.$$

Send broadcast $(\textbf{complaints-2}, sid, \Delta'_j)$ *to* $A_1, \ldots, A_n$.

16. *Upon receiving* $(\textbf{complaints-2}, sid, (A_\ell, \Delta'_\ell))$ *from* $\mathcal{F}_{\mathrm{BC}}$ *for* $\ell \in I_1$, *define the set* $\Gamma'_j := \{\ell \mid j \in \Delta'_\ell\}$.
Send broadcast $(\textbf{refutes-2}, sid, \{\zeta_{j,\ell}\}_{\ell\in\Gamma'_j})$ *to* $A_1, \ldots, A_n$.

17. *Upon receiving* $(\textbf{refutes-2}, sid, (A_\ell, \{\zeta_{\ell,i}\}_{i\in\Gamma'_\ell}))$ *for* $\ell \neq j$, *replace the old* $\zeta_{\ell,j}$ *received in step 14 if* $j \in \Gamma'_\ell$. *Then define*

$$I'_2 := \left\{\ell \in \{1,\ldots,n\} \mid g^{\zeta_{\ell,i}} = \prod_{\tau=0}^{t-1} \gamma_{\ell,\tau}^{i^\tau} \ \wedge \ h_{f(\ell)}^{\zeta_{\ell,i}} = \prod_{\tau=0}^{t-1} \delta_{\ell,\tau}^{i^\tau} \ \forall i \in \{j\} \cup \bigcup_{k=1}^{n} \Gamma'_k\right\}$$

and therewith

$$I_2 := \operatorname*{argmin}_{\substack{I\subset I'_2 \\ |I|=t}} \sum_{i\in I} i,$$

i. e. the set of the smallest t *integers in* I'_2. *Finally, go to 9.*

Voters: *Each* V_i, $i = 1, \ldots, m$, *with a* $(\textbf{public key}, sid, y)$ *message on its* `SO` *tape outputs* $(\textbf{public key}, sid, y)$.

Figure 16: Protocol $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$

Before we cite the realizability theorem, let us explain the techniques used within the protocol $\pi^{\mathrm{DL}}_{\mathrm{DKG}}$. The basic idea is to use the verifiable secret sharing (VSS) scheme of Feldman [Fel87], which overcomes the problem of sharing a secret $a_0 \in \mathbb{Z}_q$ for a prime q by using Lagrange interpolation. This in turn is based on the ideas of Shamir [Sha79]. For technical reasons, we require q to be such that $p = 2q + 1$ is a prime and there exists some cyclic group $G_q \subset \mathbb{Z}_p^*$ of order q. The idea however is simple: due to the fact that, given a polynomial $a \in \mathbb{Z}_q[x]$ of degree $t-1$ and t pairs $(x_i, a(x_i))$, $i = 1, \ldots, t$, it is possible to interpolate the polynomial $a \in \mathbb{Z}_q[x]$ *uniquely* by the formula

$$a(x) = \sum_{i=1}^{t} s_i \cdot \prod_{i \neq j} \frac{x - x_j}{x_i - x_j}.$$

Thus, by choosing $t-1$ coefficients $a_\tau \in \mathbb{Z}_q$, this defines together with a_0 a polynomial

$$a(x) = a_0 + a_1 x + \ldots + a_{t-1} x^{t-1}$$

which can be recovered by t participants using shares $s_j = a(j)$, $j = 1, \ldots, t$ and thus obtaining $a_0 = a(0)$. To check that a share s_j is valid, elements $\alpha_\tau := g^{a_\tau}$, $\tau = 0, \ldots, t-1$, are published and every participant can check the validity of s_j by proving

$$g^{s_j} \stackrel{?}{=} \prod_{\tau=0}^{t-1} \alpha_\tau^{j^\tau} = \prod_{\tau=0}^{t-1} (g^{a_\tau})^{j^\tau} = g^{\sum_{\tau=0}^{t-1} a_\tau j^\tau} = g^{a(j)}.$$

Obviously, the secrecy of this scheme relies heavily on the DDH assumption and it is thus not surprising that the security proof in [Wik05] holds if and only if the DDH assumption does. Before stating the security theorem, we give an informal description of the steps in $\pi^{\mathrm{DL}}_{\mathrm{DKG}}$.

In the first steps 1 - 3, the authority initially calls ideal functionalities. In step 4, authority A_j runs the protocol of Feldman as described above. After all authorities have done so, A_j checks in step 5 that the obtained partial secret shares $s_{\ell,j}$ are all valid. If not, it continues with step 11 with the aim to calculate the set I_1, containing all indices for which the verification in step 5 finally resists. For those, A_j then calculates its private secret share s_j as the sum of all received shares $s_{\ell,j}$ from the other authorities in I_1. In order to commit to this value s_j, authority A_j then publishes $\beta_j := g^{s_j}$ and step 8 is intended to validate all other received commitments β_ℓ for the remaining authorities A_ℓ with $\ell \in I_1$. Finally, the set I_2 contains exactly t indices, for which this validation holds true. Once given such set I_2, it is possible to set up a public key $y = g^x$, for which the private key x is only recoverable, if at least t authorities cooperate using their secret share s_j. This is the value y computed in step 9. which indeed fulfills these properties, as Lemma 4 below proves. The steps 11 - 13 and 14 - 17 finally handle the cases where validations failed. In such a case, affected authorities are required to publish their shares $s_{j,\ell}$ in order to refute that they did no cheat (steps 11 - 13) or they are requested to prove that the element β_j they published is exactly the one corresponding to their secret share s_j (steps 14 - 17). We finish our explanations with the lemma just mentioned.

Lemma 4. *Given a public key* $y := \prod_{i\in I_2} \beta_i^{\prod_{\ell\neq i} \frac{\ell}{\ell-i}}$ *as in step 9 of protocol* $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$ *then the following holds:*

(i) $y \in G_q$*, i. e.* y *has the form* $y = g^x$ *for some* $x \in \mathbb{Z}_q$*.*

(ii) There exists a polynomial $a \in \mathbb{Z}_q[z]$ *of degree* $t-1$ *such that* $s_j = a(j)$ *for all* $j = 1, \ldots, n$*, and* $a(0) = x$*.*

Proof. By step 8 we have $\beta_i = g^{s_i}$ and thus

$$y = \prod_{i\in I_2} \beta_i^{\prod_{\ell\neq i} \frac{\ell}{\ell-i}} = g^{\sum_{i\in I_2} s_i \prod_{\ell\neq i} \frac{\ell}{\ell-i}}. \tag{3.12}$$

But now, since $s_i = \sum_{r\in I_1} s_{r,i}$ and $|I_2| = t$, it holds by the LAGRANGE interpolation formula:

$$\sum_{i\in I_2} s_i \prod_{\ell\neq i} \frac{\ell}{\ell-i} = \sum_{r\in I_1} \underbrace{\sum_{i\in I_2} s_{r,i} \prod_{\ell\neq i} \frac{\ell}{\ell-i}}_{= a_r(0) \text{ as } s_{r,i}=a_r(i)} = \sum_{r\in I_1} a_r(0). \tag{3.13}$$

Denoting with $x := \sum_{r\in I_1} a_r(0) \in \mathbb{Z}_q$ we obtain by (3.12) that $y = g^x$ and this proves *(i)*.

To prove *(ii)*, set

$$a(z) := \sum_{r\in I_1} a_r(z) = \sum_{r\in I_1} \sum_{\tau=0}^{t-1} a_{r,\tau} z^\tau.$$

The polynomial a then is of degree $t-1$ and satisfies $a(0) = x$. Moreover, it holds

$$a(j) = \sum_{r\in I_1} a_r(j) = \sum_{r\in I_1} s_{r,j} = s_j$$

for $j = 1, \ldots, n$, which completes the proof. □

We now present the UC-realizability theorem obtained from [Wik05].

Theorem 6. *The* $(\mathcal{F}_{\mathrm{CRS}}, \mathcal{F}_{\mathrm{MMT}}, \mathcal{F}_{\mathrm{BC}})$*-hybrid protocol* $\pi_{\mathrm{DKG}}^{\mathrm{DL}}$ *UC-realizes* $\mathcal{F}_{\mathrm{DKG}}$*.*

Proof. A proof can be found in [Wik05, pp. 270], but for the sake of completeness, we mention the idea of the proof here. Namely, in [Wik05] an ideal process adversary $\mathcal{S}$ is constructed for every adversary $\mathcal{A}$ such that if

$$\mathrm{EXEC}_{\pi_{\mathrm{DKG}}^{\mathrm{DL}},\mathcal{A},\mathcal{Z}} \not\approx \mathrm{IDEAL}_{\mathcal{F}_{\mathrm{DKG}},\mathcal{S},\mathcal{Z}}$$

for some environment $\mathcal{Z}$, then $\mathcal{S}$ is able to break the DDH assumption, which yields a contradiction.

When evaluating the proof in [Wik05, pp. 270], one has to be aware of two differences in our presentation of the topic. Of course, the functionality $\mathcal{F}_{\mathrm{DKG}}$ in [Wik05] differs from ours, but mainly in the notation. Namely, we only provide the authorities additionally with verification keys vk_j for the threshold ELGAMAL cryptosystem, which we are interested in (see the example below). The other difference in our presentation of the protocol, beside

some error eliminations, is the use of the broadcasting functionality $\mathcal{F}_{\mathrm{BC}}$, instead of the bulletin board functionality used within the original protocol. This, however, is feasible since the broadcast is also readable from every party and the adversary, thus making no difference in the security proof. In particular, the bulletin board functionality in the original paper only makes use of the broadcasting functionality, too. □

Finally, combining Theorem 6 with Lemma 3 and Lemma 2, we obtain the following corollary.

Corollary 1. *There exists a $\mathcal{F}_{\mathrm{CRS}}$-hybrid protocol $\rho^{\mathrm{DL}}_{\mathrm{DKG}}$, which UC-realizes $\mathcal{F}_{\mathrm{DKG}}$ under the DDH assumption in G_q.*

In order to complete this section, we need to provide a threshold version of the ElGamal cryptosystem (3.10), in order to justify that we can employ the protocol $\rho^{\mathrm{DL}}_{\mathrm{DKG}}$ in $\mathcal{F}_{\mathrm{KM}}$.

Example 6 (Threshold ElGamal Encryption)**.** *Given a cyclic group $(G_q, \cdot) \subset \mathbb{Z}_p^*$ with generator g of order q with $p = 2q + 1$.*

DKG Phase: *Let each authority A_j, $j = 1, \ldots, n$, obtain a secret share $sk_j \in \mathbb{Z}_q$ of a secret $sk \in \mathbb{Z}_q$ and a verification key $vk_j \in \mathbb{Z}_q$ as done in $\pi^{\mathrm{DL}}_{\mathrm{DKG}}$.*
Note that A_j committed to sk_j by publishing $\beta_j := g^{sk_j}$.

The public key then is $pk = g^{sk}$ along with G_q and g.

The private key each authority possesses is sk_j.

Encryption: *Encryption is performed as in (3.10).*

Decryption: *Given a ciphertext $c = (c_1, c_2) = (g^r, m \cdot pk^r)$, authority A_j publishes a decryption share $ds_j := c_1^{sk_j}$ and a zero-knowledge proof[4] that $\log_g \beta_j = \log_{c_1} ds_j$ using its verification key vk_j. Let now $A_{j_1}, \ldots, A_{j_t}$ denote t authorities which passed the zero-knowledge proof. Then c can be decrypted by computing*

$$m = c_2 \cdot \left(\prod_{\nu=1}^{t} ds_{j_\nu}^{\rho_{j_\nu}} \right)^{-1},$$

where $\rho_k := \prod_{\ell \neq k} \frac{\ell}{\ell - k}$.

Note that the decryption is correct, since

$$c_2 \cdot \left(\prod_{\nu=1}^{t} ds_{j_\nu}^{\rho_{j_\nu}} \right)^{-1} = m \cdot pk^r \cdot \left(g^{r \sum_{\nu=1}^{t} sk_{j_\nu} \rho_{j_\nu}} \right)^{-1} \overset{(3.13)}{=} m \cdot g^{r \cdot sk} \cdot \left(g^{r \cdot sk} \right)^{-1} = m.$$

[4] Such a NIZK proof can work as follows: given an oracle $\mathcal{O}$, A_j can compute $a := \mathcal{O}(g^{vk_j}, c_1^{vk_j})$ and then publish the proof $p_j := (z, g^{vk_j}, c_1^{vk_j})$ with $z := vk_j + sk_j \cdot a$. Then, everybody can verify the proof by

$$\mathrm{Ver}(ds_j, p_j) = \begin{cases} \text{ok}, & g^z = g^{vk_j} \beta_j^a \wedge c_1^z = c_1^{vk_j} ds_j^a, \\ \text{fail}, & \text{otherwise.} \end{cases}$$

Now we can state the existence theorems (Theorem 7 and Theorem 8), which complete the current section dealing with the class of cryptosystems that use discrete logarithms as trapdoor to share a public key.

Theorem 7. *There exists a $\mathcal{F}_{\text{CRS}}$-hybrid protocol π_{KM} which UC-realizes $\mathcal{F}_{\text{KM}}$ for the class of DL based cryptosystems with respect to adversaries that statically block less than $n-t+1$ authorities and under the DDH assumption in G_q.*

Proof. First we decompose the ideal functionality $\mathcal{F}_{\text{KM}}$ such that $\mathcal{F}_{\text{KM}} = \mathcal{F}_{\text{DKG}} + \mathcal{F}_{\text{BB}} + \mathcal{F}_{\text{BC}}$ as in (3.9). Then, applying Theorem 5, Corollary 1 and Lemma 2 this completes the proof of the theorem since a threshold cryptosystem using a public key $y = g^x$ indeed exists, c. f. Example 6. □

This clearly implies the following result.

Theorem 8. *There exists a $\mathcal{F}_{\text{CRS}}$-hybrid protocol π_{VOTING} which UC-realizes $\mathcal{F}_{\text{VOTING}}$ for the class of DL based cryptosystems with respect to adversaries that statically block less than $n-t+1$ authorities and under the DDH assumption in G_q.*

Proof. The decomposition of the ideal functionality $\mathcal{F}_{\text{KM}}$ into $\mathcal{F}_{\text{KM}} = \mathcal{F}_{\text{DKG}} + \mathcal{F}_{\text{BB}} + \mathcal{F}_{\text{BC}}$ as in (3.9) requires us to modify the voting protocol $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ and the ideal functionality $\mathcal{F}_{\text{VOTING}}$ as mentioned at the beginning of this section. (This can also be found in the Appendix.) Then, applying Theorem 7 completes the proof. □

3.4.3.2. Cryptosystems using Factorizations

As we have seen in the previous section, it is possible to UC-realize $\mathcal{F}_{\text{DKG}}$ in the $\mathcal{F}_{\text{CRS}}$-hybrid model. This, however, implies that we can UC-realize the ideal voting functionality $\mathcal{F}_{\text{VOTING}}$ for the class of cryptosystems that use the discrete logarithm as trapdoor to share the keys in the system, i. e. where the public key is of the form $y = g^x$. As it turns out, this is much more difficult in the case where the security of the cryptosystem depends on the knowledge of the factorization of a public key $N = pq$ for large primes p and q. In the following, we refer to such cryptosystems as *factorization* (FAC) based cryptosystems.

The motivation for discussing such systems is the large number of e-voting protocols in the literature using FAC based cryptosystems, especially the PALLIER encryption we introduced in Example 3, c. f. [BFP+01, FPS01] among others (e. g. others are listed in the introduction of [PB11]). In particular, the PALLIER cryptosystem is interesting since GROTH [Gro04] explains how to modify the voting protocol $\pi_{\text{VOTING}}^{\mathcal{F}_{\text{KM}}}$ using PALLIER encryption to obtain a voting protocol that is secure against adversaries which *adaptively* corrupt authorities.

However, the setting of electronic elections requires a *threshold version* of the original PALLIER encryption in Example 3. Thus, in a first step we present such a threshold version of the PALLIER encryption from [FPS01]. Unfortunately, this is not enough since in this threshold version a trusted dealer is assumed to exist. As we will see, this dealer provides the participants with the public keys and their secret shares used to decrypt messages.

To make a voting protocol, using this encryption, trustworthy, the crucial step now is to remove the dealer from this protocol. Indeed, there exist approaches to distributedly share a RSA-key $N = pq$, first investigated by Boneh and Franklin in [BF97], which then was elaborated by Frankel et al. in [FMY98]. Afterwards, Shoup [Sho00] presented a threshold RSA scheme assuming that $N = pq$ for safe primes[5] p and q. More recently, Damgård [DK01] and Fouque [FS01] introduced protocols for threshold RSA encryption without a trusted dealer, which pursue the ideas presented in [BF97, FMY98]. We stress that the Pallier threshold encryption [FPS01] we present here, uses a RSA-modulus $N = pq$ for safe primes p, q but as we will see, this is not necessary. Therefore we briefly explain the RSA-key distribution as in [BF97]. Unfortunately this is still not enough since, the public key in the threshold version of the Pallier encryption does not only contain the modulus $N = pq$ but also additional information. Consequently, these keys have to be generated distributedly as well, which is non-trivial. Fortunately, this problem was solved very recently by Nishide and Sakurai [NS11]. Thus we finally present the main ideas of this scheme along with a discussion on approaches to UC-realize it, since this would finally yield a UC-realization of $\mathcal{F}_{\text{DKG}}$ for the threshold Pallier cryptosystem.

Threshold Pallier Cryptosystem

We present a threshold version of the Pallier encryption introduced in Example 3. This was first presented in [FPS01] and uses techniques from [Sho00]. Recall that we denote with $n \in \mathbb{N}$ the number of authorities among which the secret shares need to be distributed. In order to understand the threshold Pallier encryption we need the following lemmata.

Lemma 5. *Given $N \in \mathbb{N}$, we denote with $Q_N := \{x \in \mathbb{Z}_N^* \mid \exists y \in \mathbb{Z}_N^* : \ y^2 \equiv x \bmod N\} = \bigcup_{h \in \mathbb{Z}_N^*} h^2 \subset \mathbb{Z}_N^*$ the* **subgroup of quadratic residues** *in $(\mathbb{Z}_N^*, \cdot)$. Then, if $N = pq$ is a safe RSA modulus with $p = 2p' + 1$ and $q = 2q' + 1$, it holds:*

(i) $Q_N \subset \mathbb{Z}_N^*$ is cyclic of order $p'q'$.

(ii) $Q_{N^2} \subset \mathbb{Z}_{N^2}^*$ is cyclic of order $Np'q'$.

Proof. To prove *(i)*, observe that $Q_N \cong Q_q \times Q_p$ by the Chinese remainder theorem. Since $Q_q \subset \mathbb{Z}_q^*$ and $Q_p \subset \mathbb{Z}_p^*$ are finite subgroups of the cyclic groups $\mathbb{Z}_q^*$ and $\mathbb{Z}_p^*$, respectively, they are cyclic and since p and q are coprime, so is Q_N. Now recall that $|Q_N| = \frac{\varphi(N)}{4}$ giving $|Q_N| = p'q'$ as $\varphi(N) = 4p'q'$.

For *(ii)* the same arguments hold since $\mathbb{Z}_{q^2}^*$ and $\mathbb{Z}_{p^2}^*$ are cyclic, too. □

The preceding lemma proves that we can choose a generator of Q_N or Q_{N^2} with overwhelming probability. In fact, since $\varphi(N) \leq N - \sqrt{N}$ [SMC95, p. 9] for any composite number N it holds $\Pr(v \in Q_N) = \frac{\varphi(p'q')}{p'q'} < 1 - \frac{1}{\sqrt{p'q'}}$ and analogously $\Pr(v \in Q_{N^2}) < 1 - \frac{1}{\sqrt{Np'q'}}$.

[5] Recall that a *safe prime* p is a large prime number of the form $p = 2p' + 1$ where p' is again a prime number.

Lemma 6. *Let $N = pq$. Then, the function*

$$\Lambda : \{u < N^2 \mid u \equiv 1 \bmod N\} \ni u \longmapsto \frac{u-1}{N} \in \mathbb{Z}_N \tag{3.14}$$

is well-defined and satisfies:

$$\Lambda((1+N)^a \bmod N^2) = a \tag{3.15}$$

for all $a \in \mathbb{Z}_N$.

Proof. Let $u < N^2$ with $u \equiv 1 \bmod N$, i. e. $u = 1 + rN$ for $r < N$. We observe that $L(u) = \frac{rN}{N} = r \in \mathbb{Z}_N$ and thus L is well-defined. Now recall that

$$(1+N)^a = \sum_{k=0}^{a} \binom{a}{k} N^k = 1 + aN + N^2 \cdot \left(\sum_{k=0}^{a-2} \binom{a}{k+2} N^k \right)$$

and thus $(1+N)^a \equiv_{N^2} 1 + aN$, which yields $L((1+N)^a) = a$. □

We use this lemma in the following example to simplify the decryption process.

Example 7 (Threshold PALLIER Encryption)**.** *Let $\Delta := n!$.*

Dealing Phase: *Choose $N = pq$ with p, q being two safe primes $p = 2p'+1$ and $q = 2q'+1$ satisfying $\gcd(N, \varphi(N)) = 1$. Moreover, set $m := p'q'$. Choose $\beta \in_R \mathbb{Z}_N^*$ and $(a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ randomly and set $g := (1+N)^a \cdot b^N \bmod N^2$. Then, share the secret key $sk := \beta m$ using the SHAMIR heuristic, i. e. by choosing a polynomial $f \in_R \mathbb{Z}_{Nm}[z]$ randomly with $f(0) = sk$. Finally, choose $vk \in_R Q_{N^2} \subset \mathbb{Z}_{N^2}^*$.*

The public key then is $pk = (g, N, \theta)$ where $\theta := \Lambda(g^{m\beta} \bmod N^2) \bmod N$.
The private key each authority possesses is $sk_j := f(j)$.
The verification key each authority possesses is $vk_j := vk^{\Delta sk_j} \bmod N^2$.

Encryption: *To encrypt a message $M \in \mathbb{Z}_N$, choose $r \in_R \mathbb{Z}_N^*$ and compute the ciphertext $c := g^M r^N \bmod N^2$.*

Decryption: *Given a ciphertext $c = g^M r^N \bmod N^2$, authority A_j publishes a decryption share $ds_j := c^{2\Delta sk_j} \bmod N^2$ and a zero-knowledge proof that $\log_{c^{4\Delta}} ds_j = \log_{v^\Delta} vk_j$[6]. Now let $A_{j_1}, \ldots, A_{j_t}$ denote t authorities which passed the zero-knowledge proof. Then c can be decrypted by computing*

$$M = \Lambda\left(\prod_{\nu=1}^{t} ds_{j_\nu}^{2\rho_\nu} \bmod N^2 \right) \cdot \frac{1}{4\Delta^2\theta} \bmod N$$

where $\rho_k := \Delta \cdot \prod_{\ell \neq k} \frac{\ell}{\ell - k} \in \mathbb{Z}$.

[6] This is a NIZK proof in a cyclic group of unknown order and discussed in [FPS01, Section 3.2].

For the readers convenience, we prove that the decryption is correct. To prove this we need the following well-known properties of Carmichael's Lambda function λ, defined as $\lambda(z) = \mathrm{lcm}_{\ell=1,\ldots,k}(\lambda(p_\ell^{k_\ell}))$ where $z = \prod_{\ell=1}^{k} p_\ell^{k_\ell}$ and $\lambda(p_\ell^{k_\ell}) = p_\ell^{k_\ell-1}(p_\ell - 1)$ for $p_\ell \geq 3$ or $k_\ell \leq 2$ and $\lambda(2^{k_\ell}) = 2^{k_\ell-2}$ otherwise.

Let $N = pq$ be a safe modulus as above, then it holds:

(C1) $w^{\lambda(N)} \equiv 1 \bmod N$,

(C2) $w^{N\lambda(N)} \equiv 1 \bmod N^2$,

for all $w \in \mathbb{Z}^*_{N^2}$. This leads to the following lemma.

Lemma 7. *Given a ciphertext $c = g^M r^N \bmod N^2$ with the data defined in Example 7, then*

$$M = \Lambda\left(\prod_{\nu=1}^{t} ds_{j_\nu}^{2\rho_\nu} \bmod N^2\right) \cdot \frac{1}{4\Delta^2\theta} \bmod N,$$

i. e. the combination of the decryption shares of t authorities suffices to decrypt the ciphertext c.

Proof. Since $\Delta f(0) = \Delta m\beta = \sum_{\nu=1}^{t} sk_{j_\nu}\rho_\nu \bmod Nm$ by the Lagrange interpolation formula, we obtain

$$\prod_{\nu=1}^{t} ds_{j_\nu}^{2\rho_\nu} = \prod_{\nu=1}^{t} c^{2\Delta sk_{j_\nu} 2\rho_\nu} = c^{\sum_{\nu=1}^{t} 4\Delta sk_{j_\nu}\rho_\nu} = c^{4\Delta^2 m\beta} \bmod N^2$$

Now observe that $\lambda(N) = \mathrm{lcm}(\lambda(p), \lambda(q)) = \mathrm{lcm}(2p', 2q') = 4m$. This, together with $c = g^M r^N \bmod N^2$ and $g = (1+N)^a b^N$, finally yields

$$\prod_{\nu=1}^{t} ds_{j_\nu}^{2\rho_\nu} = g^{4\Delta^2 m\beta M} r^{4\Delta^2 m\beta N} \bmod N^2 \overset{\text{(C2)}}{=} g^{4\Delta^2 m\beta M} \bmod N^2 \overset{\text{(C2)}}{=} (1+N)^{4\Delta^2 ma\beta M} \bmod N^2. \tag{3.16}$$

Since $\theta = \Lambda(g^{m\beta} \bmod N^2) = am\beta \bmod N$ we obtain

$$\Lambda\left(\prod_{\nu=1}^{t} ds_{j_\nu}^{2\rho_\nu} \bmod N^2\right) \cdot \frac{1}{4\Delta^2\theta} \bmod N \overset{\substack{(3.16)\\(3.15)}}{=} (4\Delta^2 am\beta M) \cdot \frac{1}{4\Delta^2\theta} \bmod N = M.$$

□

A security analysis can be found in [FPS01, Theorem 1], i. e. it holds

Theorem 9. *The threshold* Pallier *encryption in Example 7 is IND-CPA secure against non-adaptive adversaries in the random oracle model.*

Groth [Gro04] proposes to use the cryptosystem of Damgård and Nielsen [DN03] to obtain a voting protocol that is secure against adaptive adversaries. However, [DN03] is again based on the threshold version of the Pallier encryption as presented above.

Shared RSA Modulus Generation

Since the distributed RSA key generation scheme by BONEH and FRANKLIN [BF97] is an important building block in the threshold version of the PALLIER encryption where the public keys are generated distributedly among the authorities, recently published by NISHIDE and SAKURAI [NS11], we give a brief explanation of it and then present the ideas behind the scheme [NS11]. Note that we only need the shared RSA modulus $N = pq$, i. e. shared public and private RSA keys e and d with $ed \equiv 1 \bmod \varphi(N)$ are not required.

On a high level, the scheme in [BF97] works as follows. Each authority A_j picks two secrets $p_j, q_j \in_R [2^{k-1}, 2^k - 1]$ for some security parameter $k \in \mathbb{N}$. The authorities then perform a **trial division test** that neither $\sum_{j=1}^{n} p_j$ nor $\sum_{j=1}^{n} q_j$ is divisible by any prime less than n, c. f. [BF97, Section 6], otherwise choosing the p_j and q_j is repeated. In a next step, the authorities agree on some large prime P such that $P > n^2 2^{2k}$. Then, using multiparty computation methods, the authorities compute

$$N := p \cdot q := \left(\sum_{j=1}^{n} p_j\right) \cdot \left(\sum_{j=1}^{n} q_j\right).$$

Here, the distributed computation of N simply uses LAGRANGE interpolation, where each authority A_j chooses two random polynomials $f_j, g_j \in_R \mathbb{Z}_P[z]$ of degree $t-1$ with $f_j(0) = p_j$ and $g_j(0) = q_j$ and another degree $2(t-1)$ polynomial $h_j \in \mathbb{Z}_P[z]$ with $h(0) = 0$ where $t = \left\lfloor \frac{n-1}{2} \right\rfloor$. Afterwards, they broadcast shares $f_j(i), g_j(i)$ and $h_j(i)$ to each authority A_i, $i = 1, \ldots, n$, and once they received all these shares, each authority interpolates the polynomial

$$\alpha(z) := \left(\sum_{i=1}^{n} f_i(z)\right) \cdot \left(\sum_{i=1}^{n} g_i(z)\right) + \sum_{i=1}^{n} h_i(z) \mod P$$

using its coefficients of $\alpha(j)$, i. e. $f_i(j), g_i(j)$ and $h_i(j)$ for $i = 1, \ldots, n$. Finally, having interpolated α, each party can compute $N = \alpha(0)$.

The authorities may again perform one (or more) trial division test that N is not divisible by any prime less than n. Once each authority has computed $N = pq$, they perform a **biprimality test** [BF97, Section 4] that N is indeed the product of two primes. During this test, nothing about the p_j and q_j is revealed. If the integer N passes this test, the protocol ends, otherwise it begins with each A_j picking integers p_j and q_j. A discussion on the performance can be found in [BF97, p. 4].

DKG for Pallier Encryption

Very recently, NISHIDE and SAKURAI [NS11] proposed a protocol to share the public key $pk = (g, N, \theta)$ and the private key $pk = \beta m$ (c. f. Example 7) among the authorities *without a trusted dealer*. This is the final step to make the threshold PALLIER encryption trustworthy to use in the setting of electronic elections. The basic idea behind the scheme [NS11] is the following. Instead of the more generic version in Example 7, [NS11] assumes a more concrete and practical instance of the cryptosystem. Namely, it has the following modifications which are important for our explanations.

Modification 1: Instead of $N = pq$ being the product of two safe primes, it is assumed that p and q are such that neither $\frac{p-1}{2}$ nor $\frac{q-1}{2}$ is divisible by any prime number less than n. This requirement is due to the fact that we still need to select generators in Q_{N^2} with high probability (c. f. Lemma 5), see [FS01, Section 2.2] for details. An adapted trial division test for such primes can be found in [NS11, Section 4.1].

Modification 2: The protocol assumes $g = N + 1$, i. e. it choses $a = b = 1$ in the tuple $(a, b) \in \mathbb{Z}_N \times \mathbb{Z}_N^*$ used within the public key generation, since this clearly simplifies the generation of g.

Modification 3: Since $N = pq$ for arbitrary primes p, q, the value m has to be defined differently. Namely, the scheme takes $m = \varphi(N)$ and uses the fact that $x^{N\varphi(N)} \equiv 1 \bmod N^2$ (since $\varphi(N^2) = N\varphi(N)$), instead of (C2).

Being aware of these modifications, the basic idea is now immediate. After having computed a RSA-modulus $N = \left(\sum_{j=1}^n p_j\right) \cdot \left(\sum_{j=1}^n q_j\right)$ as described above, each authority A_j possesses two shares p_j and q_j of it. This in turn implies that each authority holds a polynomial share of $\varphi(N)$, since

$$\varphi(N) = (p-1)(q-1) = pq - (p+q) + 1 = (N+1) - \sum_{j=1}^{n}(p_j + q_j) \mod P,$$

while the degree $(t-1)$-polynomial $\delta(z) := (\sum_{j=1}^n (f_j + g_j))(z) \bmod P$ satisfies $\delta(0) = \sum_{j=1}^n (p_j + q_j)$ and can be interpolated by any subset of t authorities. As a consequence, each authority already possesses a share of $m = \varphi(N)$. For the value $\beta \in \mathbb{Z}_N^*$, the authorities basically once more use the sharing techniques over integers with Lagrange interpolation. Concretely, each authority chooses values $\beta_j \in_R [0, KN]$ and $R_j \in_R [0, K^2N]$ randomly, with K being some security parameter, and then distributedly computes

$$\theta' := \Delta\varphi(N)\left(\sum_{j=1}^{n}\beta_j\right) + N \cdot \left(\sum_{j=1}^{n}\Delta R_i\right)$$

without revealing p_j, q_j, β_j or R_j. With $\beta := \sum_{j=1}^n \beta_j$, all parties have a polynomial sharing of $\theta = \Delta m\beta \bmod N$. See [NS11, Section 5] for details.

We finish our presentation of [NS11] here, but stress that we omitted several technical details for the sake of simplicity. For a detailed description, we refer the reader to the original paper.

About the UC-Realization of the Nishide-Sakurai Scheme

The protocol [NS11] already provides us with the necessary setup to realize the ideal DKG functionality $\mathcal{F}_{\text{DKG}}$. Let us discuss the security proofs made in [NS11, Section 6.3] and how they may be used to get a UC-realizability result. Namely, the security of the scheme described above relies on two trust assumptions [NS11] which are unfortunately non-standard but comprehensible.

Let $k \in \mathbb{N}$ be the security parameter. Then, the *RSA key generator* GE gets as input 1^k and two integers $p'', q'' \in [\ell'(2^{k-1}), \ell(2^k - 1)]$ for some $\ell = O(k)$ and $\ell' \leq \ell$. It outputs $N \leftarrow \mathrm{GE}(1^k, p'', q'')$ where $N = (p_n + p'')(q_n + q'')$ with $p_n, q_n \in_R [2^{k-1}, 2^k - 1]$ such that $p_n + p''$ and $q_n + q''$ are primes with $p_n + p'' \equiv q_n + q'' \equiv 3 \bmod 4$ and $\gcd(\frac{p_n+p''-1}{2}, \Delta) = \gcd(\frac{q_n+q''-1}{2}, \Delta) = \gcd(N, \varphi(N)) = 1$. Finally, we point out that each element $v \in Q_{N^2}$ can be represented as a tuple (v_p, v_q) with $v_p = (v_{p,1}, \ldots, v_{p,j_p})$ and $v_q = (v_{q,1}, \ldots, v_{q,j_q})$, where each $v_{p,j}$ or $v_{q,j}$ corresponds to exactly one prime number in the factorization of the order of particular generators $g_{p^2} \in Q_{p^2}$ or $g_{q^2} \in Q_{q^2}$, respectively. For details we refer to the Appendix A.2. Then we can make the following trust assumptions.

Assumption 1: Let $k \in \mathbb{N}$ be the security parameter. Let $\mathcal{B}$ be an algorithm that chooses p'', q'' and obtains a RSA modulus $N \leftarrow \mathrm{GE}(1^k, p'', q'')$. Define

$$\begin{aligned} \rho_0(k) \;:=\; & \Pr(b = 1 :\; N \leftarrow \mathrm{GE}(1^k, p'', q''), \\ & Z \in_R \mathbb{Z}_{N^2},\; b \leftarrow \mathcal{B}(1^k, N, p'', q'', Z)) \end{aligned}$$

and

$$\begin{aligned} \rho_1(k) \;:=\; & \Pr(b = 1 :\; N \leftarrow \mathrm{GE}(1^k, p'', q''), \\ & Z \in_R \mathbb{Z}_N,\; b \leftarrow \mathcal{B}(1^k, N, p'', q'', Z^N \bmod N^2)). \end{aligned}$$

We assume that $|\rho_0(k) - \rho_1(k)|$ is negligible in k for any PPT algorithm $\mathcal{B}$.

Assumption 2: Let $\mathcal{B}$ be a PPT algorithm that, on input p'', q'' and $N \leftarrow \mathrm{GE}(1^k, p'', q'')$, outputs a square $a \in Q_{N^2}$. Further let Q be the largest prime factor in $\mathrm{lcm}(p-1, q-1)$ and assume that $a \leftarrow \mathcal{B}(1^k, p'', q'', N)$ is represented as $a = (a_p, a_q)$ and a_{p,j_Q} be the number corresponding to Q. We assume that

$$\rho(k) := \Pr(a \not\equiv 1 \bmod N \;\wedge\; a_{p,j_Q} \equiv 0 \bmod Q :\; a \leftarrow \mathcal{B}(1^k, p'', q'', N))$$

is negligible in k for any PPT algorithm $\mathcal{B}$ and that $1/Q$ is negligible in k.

Obviously, the first assumption is a modification of the well-known DRC assumption (we refer to it as the *MDCR assumption*) while the second assumption assures, roughly speaking, that $\mathrm{lcm}(p-1, q-1)$ contains a large prime factor with overwhelming probability. In the proof of security, the authors of [NS11] show that their cryptosystem is IND-CPA secure in the setting of threshold cryptography if Assumption 1 holds and that the simulatability of decryption shares is assured if Assumption 2 holds. Now, the idea in the proofs is to exploit the advantage of an adversary $\mathcal{B}$ when one of the two security requirements does not hold. Using this advantage, they construct a simulator $\mathcal{S}$ which contradicts to the trust assumptions above, i. e. we have the following result.

Theorem 10. *The threshold* PALLIER *cryptosystem with the Modifications 1 - 3 above, c. f. [NS11] for a detailed description, is a threshold cryptosystem as defined in Definition 16 on page 25 under the MDRC assumption and Assumption 2 above.*

Proof. A proof can be found in [NS11, Section 6.3]. □

What remains to be shown is that the key generation part $\pi_{\text{DKG}}^{\text{Pallier}}$ of the cryptosystem above UC-realizes $\mathcal{F}_{\text{DKG}}$. Taking a look at the security proof, it seems to be reasonable that, given a simulator $\mathcal{S}$ for an adversary $\mathcal{A}$ that satisfies

$$\text{EXEC}_{\pi_{\text{DKG}}^{\text{Pallier}},\mathcal{A},\mathcal{Z}} \not\approx \text{IDEAL}_{\mathcal{F}_{\text{DKG}},\mathcal{S},\mathcal{Z}}$$

for all $\mathcal{Z}$, it is possible to prove that $\mathcal{S}$ contradicts the MDRC assumption with the arguments used within the security proof in [NS11]. This is in turn a similar approach to that of Wikström in his security proof for DL based cryptosystems as we have explained in the proof of Theorem 6. This may justify the formulation of the following conjecture.

Conjecture. *The protocol $\pi_{\text{DKG}}^{\text{Pallier}}$ UC-realizes $\mathcal{F}_{\text{DKG}}$ under the MDRC assumption.*

Unfortunately the detailed analysis of the above-mentioned approach, would go beyond the scope of this thesis but we point out that, even it is more cumbersome as in the case for DL based cryptosystems, it is reasonable that the conjecture is valid and thus it may be achievable to UC-realize $\mathcal{F}_{\text{DKG}}$ for the class of FAC based cryptosystems though.

4. Evaluating in the UC/c Framework

An important security property of protocols that arises in the setting of electronic remote elections is that of *incoercible* protocols. The term of incoercible voting schemes was first noted by Benaloh in [BT94] and can be motivated by the following descriptions. Namely, the problem is that in a remote voting protocol, a voter may obtain some kind of receipt during the voting phase, which enables him to prove that he has cast a specific vote. Since he is not required to reveal this receipt, this fact does not violate the anonymity of the protocol, but however there is an obvious ramification. Concretely, an adversary may threat the voter to cast a specific vote v^*, instead of a vote v the voter originally wanted to cast. Hence, the adversary may force the voter to reveal his receipt in order to prove that he voted v^*. We point out that such a receipt for instance may contain the randomness used by the voter to encrypt his vote, e. g. in the Pallier cryptosystem, c. f. Example 3, this may be the random value $r \in_R \mathbb{Z}_n^*$ used to compute the ciphertext $c = g^m \cdot r^n \bmod n^2$. The first attempt to bypass this drawback can be summarized in the term *receipt-freeness*, which states that a voter does not gain any receipt after the encryption. Unfortunately, this is not enough since the adversary may also gain relevant information when monitoring the traffic in a protocol execution, i. e. when he has access to the transcript of such an execution. Informally speaking, we say that a protocol is *incoercible*, if the adversary does not learn anything about the vote of a coerced voter, even if he monitors the overall message transfer between the voter and the talliers. More specific, a coercing adversary should not be able to distinguish weather a coerced party casts the vote v^* or if it is deceiving by casting the original vote v. What, however, follows from the preceding observations is that incoercible protocols are of high relevance in the setting of electronic elections, e. g. to inhibit vote buying and coercion of voters and it is thus the main intention of this chapter to discuss the possibilities to achieve incoercibility and to integrate this into the UC setting.

At a first glance, there exist three approaches to overcome the problem of coercible voters. The first was already presented by Benaloh in [BT94] and assumes the presence of a *voting booth*, which makes it impossible for a coercing adversary to observe if a voter follows its instructions or not. However, this approach does not coincide with the idea of electronic remote elections where a physically isolated trusted place such as a voting booth does not exist. The equivalent approach in the setting of remote voting is to assume an *untappable channel* between the voters and the randomizer or the talliers, which ensures that an adversary cannot tap the communication between them. For examples we refer to the references in [HNBM11]. Unfortunately, this approach contradicts to (universal) verifiability and additionally, it seems to be equivocal to require such a channel between the parties, especially in large scale elections.[1] Thus, the third approach tries to combine

[1] An untappable channel between a voter an a randomizer can be practical though. Even in large scale elections, eligible voters may obtain a tamper resistant smartcard on which the randomness is computed.

the ideas and to use *deniable cryptosystems* [CDNO96], which enable a voter to deny that he cast a certain vote in order to achieve both verifiability and incoercibility. As it turns out, this is very difficult and up to now, we are not aware of practical deniable cryptosystems which are suitable for e-voting based on homomorphic encryption.

Since already a lot of protocols in the literature claim to be incoercible, c. f. [JCJ05, BT94, CCM08] among others, while only providing informal definitions, it is necessary to subsume the present ideas, c. f. [BT94, DKR06] among others, and to present a formal description of incoercibility, which fits well into the UC setting. Therefore, we first present the UC/c framework by UNRUH and MÜLLER-QUADE [UMQ10] (c. f. the full version [UMQ09], too) in Section 4.1. Within this paper, the authors propose a formalization of incoercibility by extending the UC framework of CANETTI. This is well suited for our contributions to evaluate security of voting protocols in the UC framework, since the communication model remains the same. What changes is that the term of the realization of a protocol π is slightly modified by introducing an additional party $\mathcal{D}$ to the execution of π which is called the *deceiver*. Finally, it is necessary to adapt the corruption model and to prove that the important property of universal composition remains true even in the modified framework. Section 4.2 is devoted to the discussion on how to apply this framework to the setting of electronic elections, which is done by first defining incoercibility for voting protocols and then proving that this term is already implied if a voting protocol π *UC/c-realizes* the voting functionality $\mathcal{F}_{\text{VOTING}}$. Unfortunately, the UC/c-realization turns out to be hard to achieve. A first attempt to overcome this drawback is to study deniable cryptosystems, first presented by CANETTI et al. [CDNO96]. Thus, Section 4.3 provides a discussion on this class of cryptosystems.

4.1. The UC/c Framework

The *Composable Incoercibility framework* (UC/c) [UMQ10] by UNRUH and MÜLLER-QUADE, is an approach to formalize incoercible multi-party protocols. The intuition behind this framework is to define what it means that an adversary $\mathcal{A}$, coercing a party P to perform some specific action, can distinguish weather P follows its instructions and thus performing this action (we then call P an *obeying* party) or not (P is *deceiving*). As it turns out, this natural approach cannot be achieved, e. g. in electronic voting protocols the adversary learns the tally and might always distinguish the distributions when P casts some vote v^* forced by $\mathcal{A}$ or casting the original vote v, c. f. [UMQ10, p. 413]. The authors of [UMQ10] thus propose to require that this advantage of $\mathcal{A}$ should be bounded above by the advantage the adversary could gain from the information in the ideal world, i. e. the information which is a priori legally given to him. To formalize this, an additional party $\mathcal{D}$ is added to the ideal world execution of $\text{IDEAL}_{\mathcal{F}}$ for some ideal functionality $\mathcal{F}$, called the *deceiver*. The security term, which arises, compares this modified ideal world execution with the execution of some protocol π, but now requiring additionally to the UC-realizability in the UC framework that in the real world execution there exists a *deceiver simulator* such that any environment $\mathcal{Z}$ cannot distinguish the just explained real

world execution of π and the ideal world execution of $\mathcal{F}$.

Before we formally define the term of UC/c-realizability, we introduce the deceiver just mentioned as an additional PPT ITM $\mathcal{D}$ with `identity` *dec* and adapt the execution of a multi-party protocol π as follows.

Definition 32. *An* **execution of a protocol** π *in the UC/c framework is defined as the system* $(\mathcal{Z}, C_{\text{EXEC}}^{\pi,\mathcal{A},\mathcal{D}})$ *of ITMs with*

- *a PPT ITM* $\mathcal{Z}$*, called the* **environment**,
- *a PPT ITM* $\mathcal{A}$*, called the* **adversary**,
- *a PPT ITM* $\mathcal{D}$*, called the* **deceiver** *and*
- *a PPT ITM* π *for the given protocol.*

Here, the control function $C_{\text{EXEC}}^{\pi,\mathcal{A},\mathcal{D}}$ *is defined as follows. Initially,* $\mathcal{Z}$ *is invoked with some initial input* $z \in \{0,1\}^*$ *and a security parameter* $k \in \mathbb{N}$*. Then:*

(C1) The first ITM invoked by $\mathcal{Z}$ *is the deceiver* $\mathcal{D}$ *with* `identity` *dec followed by the adversary* $\mathcal{A}$*. All other ITMs invoked by* $\mathcal{Z}$ *have the code of* π *and the same* SID*.*

(C2) When the adversary $\mathcal{A}$ *is activated, it can pass output to* $\mathcal{Z}$ *and additionally perform one of the following activities.*

- *(C2a)* $\mathcal{A}$ *can deliver a message* m *to a (sub-)party.*
- *(C2b)* $\mathcal{A}$ *can corrupt a (sub-)party if it is instructed by* $\mathcal{Z}$ *to do so via a corresponding corrupt message from* $\mathcal{Z}$*.*

(C3) If a (sub-)party of π *is activated (by* $\mathcal{Z}, \mathcal{A}$ *or another (sub-)party), then it can send messages to* $\mathcal{A}$ *and pass input or output to other (sub-)parties with the same* SID*. Additionally, parties of* π *can pass output to* $\mathcal{Z}$*.*

(C4) When the deceiver $\mathcal{D}$ *is activated, it can communicate with* $\mathcal{Z}$ *and* $\mathcal{A}$ *without any restrictions. Moreover,*

- *(C4a)* $\mathcal{D}$ *can deliver a message* m *to a (sub-)party.*
- *(C4b)* $\mathcal{D}$ *can send a deception request to a (sub-)party if it is instructed by* $\mathcal{Z}$ *to do so via a corresponding deception message from* $\mathcal{Z}$*.*

Within the UC framework, we used the corruption model, where a corrupted party sent its entire state to the adversary and was from there on controlled by it. Here, we need to refine this corruption model as in [UMQ10, p. 417], since the deceiver was added to the execution of the protocol.

Definition 33 (Corruption Model in the UC/c framework)**.** *A party P in the execution of a protocol π can be either* **uncontrolled**, **corrupted** *or* **deceiving**. *An uncontrolled party can become corrupted, if it receives a corruption request from $\mathcal{A}$ and it can become deceiving, if it receives a deception request from $\mathcal{D}$. When a party becomes corrupted or deceiving, it sends its entire state to the adversary or the deceiver, respectively, and is from that time on controlled by $\mathcal{A}$ or $\mathcal{D}$, respectively. The important restriction is, however, that if a corrupted party receives a deception request, it does* not *forward this to the adversary. Moreover, a deceiving party cannot be corrupted.*[2]

The definition of the adapted distribution ensemble $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D},\mathcal{Z}}$ is straightforward.

Definition 34. *Given some input $z \in \{0,1\}^*$ and security parameter $k \in \mathbb{N}$, we define $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D},\mathcal{Z}}(k,z) \in \{0,1\}$ to be the output of the system $(\mathcal{Z}, C_{\mathrm{EXEC}}^{\pi,\mathcal{A},\mathcal{D}})$ (c. f. Definition 32). Naturally, this binary variable induces a binary distribution ensemble $\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D},\mathcal{Z}}$ defined through*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D},\mathcal{Z}} := \bigcup_{k\in\mathbb{N}} \bigcup_{z\in\{0,1\}^*} \mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D},\mathcal{Z}}(k,z).$$

We are now in the position to define the term of UC/c-emulation.

Definition 35. [UMQ10, Definition 2] *Let π and ρ be PPT protocols. We say that π* **UC/c-emulates** *ρ, if for any PPT deceiver $\mathcal{D}$ there exists a PPT deceiver $\mathcal{D}_S$ such that for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_S$ such that*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D}_S,\mathcal{Z}} \approx \mathrm{EXEC}_{\rho,\mathcal{A}_S,\mathcal{D},\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

Analogously, we define the term of UC/c-realization as follows.

Definition 36. *Let π be a PPT protocol and $\mathcal{F}$ an ideal functionality. We say that π* **UC/c-realizes** *$\mathcal{F}$, if π UC-emulates $\mathrm{IDEAL}_{\mathcal{F}}$, i. e. for any PPT deceiver $\mathcal{D}$ there exists a PPT deceiver $\mathcal{D}_S$ such that for any PPT adversary $\mathcal{A}$ there exists a PPT adversary $\mathcal{A}_S$ such that*

$$\mathrm{EXEC}_{\pi,\mathcal{A},\mathcal{D}_S,\mathcal{Z}} \approx \mathrm{EXEC}_{\mathrm{IDEAL}_{\mathcal{F}},\mathcal{A}_S,\mathcal{D},\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

Let us recall Definition 33, which defines how corruption and deception in the UC/c framework is modeled. In particular, it allows a corrupted party P to get deceiving. As UNRUH et al. [UMQ10, p. 419] notice, it is very hard to achieve a UC/c-emulation, if we allow this kind of corruption model that UNRUH et al. call *bad-guy coercions*, since a coerced party P preliminarily gets corrupted and thus is a "bad guy". The simple reason is that a corrupted party P completely follows the adversaries instructions, which makes it very difficult for the deceiver to convince the adversary that P still follows its

[2] A motivation of these restrictions and the overall corruption model can be found in [UMQ10, pp. 418].

instructions in a later step. For example, the adversary may force P to reveal some secrets (e. g. the randomness $r \in \mathbb{Z}_n^*$ in the Pallier encryption) and afterwards, party P has no possibility to deceive since the only parameters used by P consist of some public data along with the revealed secret. This is why it can be reasonable to disallow a corrupted party to get deceiving which results in the term of *good-guy coercions* and we think that, if even possible, this type of coercion is the only possibility to achieve UC/c secure voting protocols.

Before we state the adapted universal composition theorem for the UC/c framework, we remind the reader that we introduced a special adversary in Lemma 1 on page 35, called the *dummy adversary* $\mathcal{A}_D$, which simply forwards messages from the environment to parties and vice versa. What immediately comes up is the question, if we can additionally restrict ourselves to a *dummy deceiver* $\mathcal{D}_D$, which is defined similarly. Indeed, this is the case, as the following lemma shows, c. f. [UMQ10, Lemma 4].

Lemma 8. [UMQ10, Lemma 4] *Let π and ρ be PPT protocols. Then π UC/c-emulates ρ according to Definition 35, if and only if there exists a PPT deceiver $\mathcal{D}_S$ and a PPT adversary $\mathcal{A}_S$ such that*

$$\mathrm{EXEC}_{\pi,\mathcal{A}_D,\mathcal{D}_S,\mathcal{Z}} \approx \mathrm{EXEC}_{\rho,\mathcal{A}_S,\mathcal{D}_D,\mathcal{Z}}$$

for all PPT environments $\mathcal{Z}$.

The immediate consequence is that we only have to provide a deceiver simulator $\mathcal{D}_S$ for the generic dummy deceiver $\mathcal{D}_D$. Hence, such a machine $\mathcal{D}_S$ can be seen as some kind of generic deception strategy for a protocol π. The following theorem completes our brief introduction of the UC/c framework.

Theorem 11. [UMQ10, Theorem 5] *Let π, ρ and ρ' be PPT multi-party protocols such that ρ UC/c-emulates ρ' and both ρ and ρ' are subroutine respecting. Then protocol $\pi^{\rho'/\rho}$ UC/c-emulates protocol π.*

4.2. Applications to Electronic Voting

The present section bridges the gap between the security notion of the UC/c framework and incoercible voting protocols. To achieve this, we need to find a formal definition of an incoercible voting protocol. The final definition was basically already given in [UMQ10, Definition 9], while we do not restrict the adversary to only coerce a *single* voter V_i to cast a vote v^* but we let him be able to control an arbitrary subset of $\{V_1, \ldots, V_m\}$. The final definition then has the intuition that a voting protocol is incoercible if the coercing adversary's advantage in deciding if the corrupted parties are deceiving or obeying is bounded above by the advantage that he could distinguish only using the tally. Finally, we will present in Theorem 12 the crucial result which is due to [UMQ10, Theorem 10] with the exception that we consider adversaries that may coerce multiple voters.

However, before we get concrete, we need to slightly modify the voting functionality $\mathcal{F}_{\text{VOTING}}$, where we use the extended version from the Appendix A.1, c. f. Definition 41.

Let $\mathcal{T}$ denote a set of possible tallies, e. g. $\mathcal{T} = \mathcal{V}$, $\mathcal{T} = \{\mathcal{V} \longrightarrow \mathbb{N}_0\}$ or $\mathcal{T} = \mathcal{V} \times \mathbb{N}_0$. This is only a formalization of how to publish the final tally.

Definition 37. *The ideal voting functionality $\mathcal{F}^{\text{tally}}_{\text{VOTING}}$, using an efficiently computable tallying function* $\text{tally} : (\mathcal{V} \cup \{\bot\})^m \longrightarrow \mathcal{T}$*, is defined as follows.*

Parties *: adversary $\mathcal{S}$, $V_1, \ldots, V_m$ (voters), $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor)*

- *Upon receiving* $(\textbf{vote}, sid, (V_i, v))$ *from V_i store it and send* $(\textbf{vote}, sid, V_i)$ *to $\mathcal{S}$. Ignore subsequent* **vote** *messages from V_i with* SID *sid.*
- *Upon receiving* $(\textbf{no-block}, sid, V_i)$ *from $\mathcal{S}$, check if some* $(\textbf{vote}, sid, (V_i, v))$ *has been stored. In that case store* $(V_i, v_i := v)$*.*
  Ignore subsequent **no-block** *messages for V_i with* SID *sid from $\mathcal{S}$.*
- *Upon receiving* $(\textbf{result}, sid)$ *from T compute the final result* $\mathfrak{t} = \text{tally}(\{v_i\}_{i=1}^m)$ *($v_\ell := \bot$ for which no vote was received), store it and send* $(\textbf{result}, sid, \mathfrak{t})$ *to $\mathcal{S}$.*
- *Upon receiving* $(\textbf{deliver-result}, sid, P)$ *from $\mathcal{S}$, send* $(\textbf{result}, sid, \mathfrak{t})$ *to P, where* $P := \{A_1, \ldots, A_n, T\}$*.*

Figure 17: Functionality $\mathcal{F}^{\text{tally}}_{\text{VOTING}}$

Before we present the definition of an incoercible voting scheme, we fix some notation, c. f. [UMQ10, p.423]. Let $\mathfrak{C} \subseteq \{V_1, \ldots, V_m\}$ be a set of voters with $|\mathfrak{C}| = m' \leq m$ and let $\mathfrak{v} \in (\mathcal{V} \cup \{\bot\})^{m'}$ be the tuple of votes for the voters in $\mathfrak{C}$. Finally, we fix a probability distribution $\mathcal{B}$ on $(\mathcal{V} \cup \{\bot\})^{m-m'}$.

Definition 38. *We use the following notations.*

1. *Given $\mathfrak{v} \in (\mathcal{V} \cup \{\bot\})^{m'}$, let $\mathcal{B}_\mathfrak{v}$ denote the probability distribution over $(\mathcal{V} \cup \{\bot\})^m$ that chooses the votes $v_i \in \mathcal{V} \cup \{\bot\}$ for voters $V_i \in \mathcal{V} \setminus \mathfrak{C}$ according to $\mathcal{B}$ and as in $\mathfrak{v}$ otherwise.*[3]
2. *We denote by* $\text{tally}(\mathcal{B}_\mathfrak{v})$ *the random variable that outputs values $\mathfrak{t} \in \mathcal{T}$ with inputs chosen according to $\mathcal{B}_\mathfrak{v}$.*
3. $\text{Adv}_{\text{ideal}}(\mathcal{B}, \mathfrak{v}) := \max_{\mathfrak{v}^*} \sum_{v \in (\mathcal{V} \cup \{\bot\})^m} |\mathcal{B}_\mathfrak{v}(v) - \mathcal{B}_{\mathfrak{v}^*}(v)|$*, where the maximum is taken over all* $\mathfrak{v}^* \in (\mathcal{V} \cup \{\bot\})^{m'}$*.*
4. *A* **voting adversary** *$\mathcal{A}$ for $\mathfrak{C}$ is an adversary that controls the parties in $\mathfrak{C}$. Moreover, $\mathcal{A}$ may start the tally and when T outputs $\mathfrak{t}$, this value is given to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.*

[3] This means, that already $\mathcal{B}_\mathfrak{v}((\mathcal{V} \cup \{\bot\})^{m-m'} \times \{\mathfrak{v}\}) = 1$, i. e. all other probabilities of points are zero.

5. *We denote by* $\Pr_{\text{obey}}(\mathcal{A}, \mathcal{B})$ *the probability that* $\mathcal{A}$ *outputs* 1 *in the case that the parties in* $\mathfrak{C}$ *follow its instructions and all other parties follow the protocol honestly with inputs chosen according to* $\mathcal{B}$*.*

6. *We denote by* $\mathfrak{d}(\mathfrak{v}, \mathfrak{C})$ *some program code for the parties in* $\mathfrak{C}$*. Informally, this is the deception strategy for the parties in* $\mathfrak{C}$*.*

7. *We denote by* $\Pr_{\text{deceive}}(\mathcal{A}, \mathfrak{d}, \mathcal{B})$ *the probability that* $\mathcal{A}$ *outputs* 1 *in the case that the parties in* $\mathfrak{C}$ *follow the instructions in* $\mathfrak{d}$ *and all other parties follow the protocol honestly with inputs chosen according to* $\mathcal{B}$*.*

8. *We denote by* $\text{Tally}_{\text{deceive}}(\mathcal{A}, \mathfrak{d}, \mathcal{B})$ *the output of* T *in the case that the parties in* $\mathfrak{C}$ *follow the instructions in* $\mathfrak{d}$ *and all other parties follow the protocol honestly with inputs chosen according to* $\mathcal{B}$*.*

We are now in the position to define the notion of an incoercible voting protocol. We stress that we forgo the definition of a voting scheme in general and we require instead the protocol to UC-realize $\mathcal{F}_{\text{Voting}}^{\text{tally}}$ which is possible due to Corollary 1. However, this implies that the protocol involves the required parties for the following definition, c. f. [UMQ10, Definition 9].

Definition 39. *Let* π *be a voting protocol depending on some security parameter* $k \in \mathbb{N}$ *that UC-realizes* $\mathcal{F}_{\text{Voting}}^{\text{tally}}$*. Then* π *is* **incoercible***, if there exists a deception strategy* $\mathfrak{d}$ *such that for every PPT voting adversary* $\mathcal{A}$*, every subset* $\mathfrak{C} \subseteq \{V_1, \ldots, V_m\}$ *with* $|\mathfrak{C}| = m'$*, every* $\mathfrak{v} \in \mathcal{V}^{m'}$ *and every efficiently sampleable distribution* $\mathcal{B}$ *the following holds.*

1. *(The deception strategy casts the right vote.)* $\text{Tally}_{\text{deceive}}(\mathcal{A}, \mathfrak{d}, \mathcal{B})$ *and* $\text{tally}(\mathcal{B}_{\mathfrak{v}})$ *are computationally indistinguishable.*

2. *(The adversary cannot distinguish between obeying and deceiving parties.) For some negligible function* μ *it holds* $|\Pr_{\text{obey}}(\mathcal{A}, \mathcal{B}) - \Pr_{\text{deceive}}(\mathcal{A}, \mathfrak{d}, \mathcal{B})| \leq \text{Adv}_{\text{ideal}}(\mathcal{B}, \mathfrak{v}) + \mu$*.*

The following theorem bridges the gap between UC/c-emulation and incoercible voting schemes.

Theorem 12. *Let* π *be a PPT protocol that UC/c-realizes* $\mathcal{F}_{\text{Voting}}^{\text{tally}}$ *with static corruption and deception. Then* π *is an incoercible voting scheme due to Definition 39.*

Proof. A straightforward proof can be found in [UMQ10, pp. 424]. □

4.3. Deniable Encryption

As we have seen in the preceding section, a protocol π that UC/c-realizes $\mathcal{F}_{\text{Voting}}^{\text{tally}}$ with static corruption and deception is at the same time an incoercible voting scheme. Unfortunately, it turns out to be very difficult to achieve such a UC/c-realization $\mathcal{F}_{\text{Voting}}^{\text{tally}}$, even if we consider good-guy coercions only. This drawback was already mentioned by Unruh

and MÜLLER-QUADE in [UMQ10, p. 426] and they claim that it would be necessary to involve novel cryptographic techniques.

A first attempt to overcome this drawback is to study *deniable cryptosystems*, first presented by CANETTI et al. [CDNO96] which are, informally speaking, public-key cryptosystems (E, D), where randomness in the encryption process is used to fake an adversary by revealing some different randomness. More specific, consider a public-key cryptosystem with an encryption function

$$E : \mathcal{M} \times \mathcal{R} \longrightarrow \mathcal{C}$$

for some finite set $\mathcal{R}$ and a ciphertext $c = E_{pk}(m, r)$ being the encryption of a plaintext $m \in \mathcal{M}$ using some randomness $r \in_R \mathcal{R}$. The key idea in defining deniable cryptosystems is to assume the existence of a faking algorithm

$$\text{fake} : \mathcal{M} \times \mathcal{M} \times \mathcal{C} \times \mathcal{R} \longrightarrow \mathcal{R}$$

which, given as input, the real plaintext m, a fake message m', the ciphertext $c = E_{pk}(m, r)$ and the randomness r, outputs a value $r' := \text{fake}(m, m', c, r) \in \mathcal{R}$ such that $c = E_{pk}(m', r')$. If such a faking algorithm exists, then (E, D) is a *sender-deniable* cryptosystem. For a detailed description, we refer to [CDNO96]. Beside this class, there exist *receiver-deniable* cryptosystems, too. However, important for our purposes are sender-deniable cryptosystems, as the voters are the only possibly coerced parties.

In the preceding chapter we introduced two cryptosystems which are well-suited for the voting protocol $\pi^{\mathcal{F}_{\text{KM}}}_{\text{VOTING}}$, c. f. Section 3.3.2, that requires homomorphic threshold encryption. Namely, the encryption functions E_{EG} and E_{PALLIER} for the ELGAMAL and the PALLIER cryptosystem, respectively, fulfill a certain homomorphism property. We recall: given a plaintext $m \in \mathcal{M}$ and randomness $r \in \mathcal{R}$, they compute the ciphertext $c \in \mathcal{C}$ as

$$\begin{aligned} E_{\text{EG}}(m, r) &:= c := (g^r, my^r), \\ E_{\text{Pallier}}(m, r) &:= c := g^m r^N \bmod N^2, \end{aligned}$$

for public values $y = g^x$ and $N = pq$, respectively. Unfortunately, these cryptosystems fail to guarantee incoercibility since an adversary can easily coerce a voter to reveal a receipt of his vote by demanding the randomness r. In both schemes, the sender (which is the voter) is not able to deceive the eavesdropping adversary in order to make him believe that the ciphertext c belongs to another vote m'. The reason is that the voter usually does not have more information about a certain trapdoor (e. g. factorization, DL problem, quadratic residue problem, higher residue problem etc.) than the adversary has. Consequently, he has no chance to deceive him, otherwise he would be able to efficiently compute the trapdoor, which is a contradiction.

Although there already exist different deniable cryptosystems in the literature, c. f. [CDNO96, Ibr09, KKK08, HB09, HNBM11] among others, most of them are not practically useful in electronic remote election protocols for several reasons. First, most of the schemes provide only probabilistic correctness, i. e. the decryption of a certain ciphertext is only possible for a receiver with a certain probability $\rho < 1$. Obviously, these schemes are useless

in the setting of electronic elections since the authorities need to be able to compute the correct tally with probability one. The second drawback of present deniable cryptosystems is that most of them are bit-encryption schemes, i. e. the encryption function only encrypts a single bit. Although this disadvantage can be overcome, the main problem is that they produce very large ciphertexts in comparison to the plaintext (which is only one bit).

In the following paragraphs we describe interesting approaches leading to deniable cryptography and discuss how to use them inside of e-voting protocols. The first approach is to encrypt messages using XOR operations on the plaintext, i. e. by calculating $c = m \oplus r$ for some randomness $r \in \{0,1\}^\ell$, an example can be found in [HNBM11]. The intention here is that it is very easy for the sender to reveal a fake randomness $r' \in \{0,1\}^\ell$ and a fake message $m' \in \{0,1\}^\ell$ such that they satisfy $c = m' \oplus r'$. The security and the deniability property of the cryptosystem in [HNBM11] are based on the quadratic residue assumption, but the correctness probability of the system is again only overwhelming but not equal to one. A similar approach can be found in [Ibr09], but the big advantage of this scheme is that the receiver always computes the correct plaintext. Unfortunately, these schemes are not appropriate for e-voting protocols based on homomorphic encryption, but they could be used for e-voting based on mix-nets though. Another interesting approach is due to Dürmuth et al. [DF11]. They propose a generic deniable encryption scheme which uses *sampleable* cryptosystems, c. f. [DF11, Definition 3.1], as building block in their scheme and explain how to gain a deniable encryption scheme from it. The question is, if there exist homomorphic cryptosystems which are sampleable and the answer is yes. Namely, the authors show that the Goldwasser-Micali cryptosystem, where a ciphertext of a bit $b \in \{0,1\}$ is computed as

$$c = E_{\text{G-M}}(b, r) := g^b r^2 \bmod N$$

with $N = pq$, is sampleable but unfortunately, the homomorphic property is not well-suited for voting protocols, since we only have $E_{\text{G-M}}(b_1, r_1) \cdot E_{\text{G-M}}(b_2, r_2) = g^{b_1 \oplus b_2}(r_1 r_2)^2 \bmod N$, which is not useful. Another drawback is that the receiver again is only able to decrypt the encrypted bit $b \in \{0,1\}$ correctly with probability $1/2 + 1/(5\sqrt{n})$, c. f. [DF11, Lemma 4.2]. Nevertheless, this scheme encourages to intensify the research for suitable deniable cryptosystems.

Let us briefly discuss how to apply a deniable cryptosystem (E, D) with a faking algorithm in an execution of a protocol π that UC-realizes $\mathcal{F}_{\text{VOTING}}^{\text{tally}}$ and where the votes v_i are sent encrypted as $x_i := E_{pk}(v_i, r_i)$ by a voter V_i to the tallier within the UC/c framework. The required deceiver-simulator $\mathcal{D}_S$ in the definition of UC/c-emulation (c. f. Definition 36) needs to provide a coerced voter V_k with a deceiving strategy such that the adversary cannot tell whether V_k is obeying or deceiving. But this however is immediate when V_k can invoke fake : $\mathcal{M} \times \mathcal{M} \times \mathcal{C} \times \mathcal{R} \longrightarrow \mathcal{R}$ to produce another random value $\hat{r}_i \in \mathcal{R}$, i. e. he computes

$$x_i := E_{pk}(\hat{v}_i, \hat{r}_i) \text{ with } \hat{r}_i := \text{fake}(v_i, \hat{v}_i, x_i, r_i)$$

and then reveals $(\hat{v}_i, \hat{r}_i)$ to the adversary, being not able to distinguish. This motivates the usage of deniable cryptosystems but in the proposed setting of homomorphic cryptosys-

tems there exists a series of crucial drawbacks. First, a voter V_i needs to prove that he has cast a vote $v_i \in \mathcal{V}$ with a NIZK proof, e. g. using a Σ-protocol, while the proof additionally may contain receipts of the vote which can be used by the adversary to coerce V_i (using a Σ-protocol this is for instance the oracle answer a_i in a proof $p = (a_i, z_i, aux_i)$). Secondly, as we have mentioned above, we are not aware of any practically useful cryptosystem which is both, deniable and homomorphic.

Finally we may have a look at the following approach: consider the deniable encryption scheme in [Ibr09], which provides a deniable bit-encryption scheme (E_d, D_d) with an error-less decryption, which can be extended to a multi-bit encryption scheme. With each such cryptosystem, we can construct a deniable cryptosystem for the e-voting setting. Namely, we require each voter V_i, which wants to cast a vote v_k to do proceed as follows.

1. V_i encrypts all votes $\mathcal{V} = \{v_1, \ldots, v_\ell\}$ and obtains $x_i^j := E_{pk}(v_j, r_j)$.
2. V_i computes a proof p_j for each x_i^j, $j = 1, \ldots, \ell$.
3. V_i chooses some permutation $\pi : \{1, \ldots, \ell\} \longrightarrow \{1, \ldots, \ell\}$ at random.
4. V_i encrypts $\pi(k) = \hat{k}$ using E_d.
5. The voter then sends $C := (\mathcal{C}, \mathfrak{k})$ with $\mathfrak{k} := E_d(\hat{k})$ and $\mathcal{C} := (x_i^{\pi(1)}, \ldots, x_i^{\pi(\ell)})$ to an authority A.
6. The authority A can compute $\hat{k} = D_d(\mathfrak{k})$ and knows that $x_i^{\hat{k}}$ is the vote the voter originally wanted to cast.

Obviously, V_i is now in the position to deceive a coercing adversary, since it may reveal receipts used within the encryption process of E_d to the adversary such that $E_d(k') = \mathfrak{k}$ and k' is the index corresponding to the vote $v_{k'}$ the adversary forced V_i to vote for. Unfortunately, this idea fails to guarantee deniability in the setting, where we require a threshold of t authorities to cooperate in order to decrypt a product of encrypted votes. Namely, an adversary which coerces a party may also corrupt a single authority A. Even if we require that t authorities need to cooperate in order to compute $\hat{k}$ and thus to decrypt $\mathfrak{k}$, the adversary may instruct an authority A to act honestly in order to obtain the value $\hat{k}$ which contradicts the deniability property. Therefore we need that the authorities can compute shares of $\mathfrak{k}$ such that these shares do not leak any information about the $\hat{k}$ but can be used to combine a bunch of broadcast ciphertexts $C := (\mathcal{C}, \mathfrak{k})$ in some sense such that this combination finally can be decrypted by a threshold of t authorities. Unfortunately, we do not see how this can be achieved with the scheme [Ibr09] and claim that this requires new and yet unstudied techniques.

5. Conclusions

Within this thesis, we gave a brief overview over techniques used in recent electronic voting schemes. It turned out that only two of these are of current interest (mix-net based and homomorphic voting schemes) and that each of them has several advantages and disadvantages making it difficult to decide which approach may lead to secure, practical and efficient voting schemes.

We also mentioned that there exists a large and unmanageable series of voting schemes in the literature, where many of them provide (partially) new ideas while they often do not give security proofs. This makes it hard to compare schemes or decide which is more or less secure. To overcome this problem, Groth [Gro04] proposed to evaluate new voting schemes in the UC framework, which we presented within this thesis. The UC framework has the big advantage that secure protocols can be executed concurrently and thus within more complex settings.

It turned out that the class of homomorphic voting schemes securely realizes an ideal voting functionality $\mathcal{F}_{\text{Voting}}$ in the UC framework. Due to this fact we concentrated on homomorphic voting within the second half of the thesis and presented the protocol and security proof of [Gro04]. We presented necessary techniques used to realize the ideal voting functionality $\mathcal{F}_{\text{Voting}}$. Concretely we defined homomorphic threshold voting and Σ-protocols made non-interactive by a Fiat-Shamir heuristic and provided security proofs for these proofs being zero-knowledge and non-malleable (Theorem 3). Moreover we showed how to UC-realize the message board functionality $\mathcal{F}_{\text{KM}}$, used within the original paper [Gro04], in order to obtain a voting protocol that $\mathcal{F}_{\text{Voting}}$ with only having access to a common reference string. It turned out that we only obtain this result for the class of discrete logarithm based cryptosystems, c. f. Theorem 6 and Corollary 1. In order to obtain this result, we used the ElGamal encryption and thus got a practically useful voting protocol which can be proven secure in the UC framework, c. f. Section 3.4.3.1. For the class of cryptosystems that use factorization as trapdoor such as the Pallier encryption, it turned out that it is much more difficult to obtain such a result. However, we claimed that this is possible though, c. f. Conjecture 3.4.3.2, since all necessary schemes, needed to share a public key of the threshold version of Pallier encryption, are known to exist and a proof of UC-security seems feasible.

Incoercibility is an important security property for electronic remote voting in order to prevent vote buying or coercions while there is up to now no consensus on a definition of this term. To overcome this problem we introduced the UC/c framework [UMQ10] by Unruh and Müller-Quade, which turned out to be well-suited for the setting of electronic remote voting, c. f. Theorem 12. Unfortunately a UC/c-realization of $\mathcal{F}_{\text{Voting}}$ is difficult

to achieve. We thus presented the paradigm of deniable encryption in Section 4.3, which is suitable for incoercible protocols while we however need a deniable cryptosystem that is homomorphic in order to use this within the voting protocol presented in Section 3.3. This is still an open problem, since so far we are not aware of any practical encryption which is both homomorphic and deniable. Finally, we sketched the idea how to use a deniable bit-encryption scheme such as [Ibr09] in order to obtain a deniable voting scheme. This was done by simply encrypting each possible vote and sending this information together with an encrypted index – using the deniable bit-encryption scheme – that indicates which of the ciphertext contains the correct vote. Unfortunately, this does not yield a deniable voting scheme for homomorphic threshold encryption, since an adversary may also corrupt an authority, which provides him with the correct index.

A. Appendix

A.1. Modified Protocols and Functionalities

Within Section 3.4 we gave a discussion on how to UC-realize the ideal functionality $\mathcal{F}_{\text{KM}}$. To provide this realization it is necessary to add an additional party T to the ideal functionalities $\mathcal{F}_{\text{VOTING}}$, $\mathcal{F}_{\text{KM}}$ and to the protocol $\pi^{\mathcal{F}_{\text{KM}}}_{\text{VOTING}}$ defined in Sections 3.2.3, 3.3.1 and 3.3.2, respectively. Furthermore, we decompose the functionality $\mathcal{F}_{\text{KM}}$ such that

$$\mathcal{F}_{\text{KM}} = \mathcal{F}_{\text{DKG}} + \mathcal{F}_{\text{BB}} + \mathcal{F}_{\text{BC}}. \tag{A.1}$$

Additionally, we need to restrict the functionality $\mathcal{F}_{\text{KM}}$ to send the $(\textbf{post}, sid, (A_i, m))$ messages as delayed output to the authorities and not immediately to all $A_1, \ldots, A_n$.

The modified Message Board

Definition 40. *The modified ideal message board functionality $\mathcal{F}^{\text{mod}}_{\text{KM}}$ is defined as follows.*

Parties *: adversary $\mathcal{A}$, $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor)*

- *Generate keys $(pk, vk_1, \ldots, vk_n, sk_1, \ldots, sk_n)$ for the homomorphic threshold cryptosystem as in Definition 17.*
  Send $(\textbf{public key}, sid, pk)$ to $V_1, \ldots, V_m, A_1, \ldots, A_n$ and $\mathcal{A}$.
  Send $(\textbf{verification keys}, sid, (vk_1, \ldots, vk_n))$ to $A_1, \ldots, A_n$ and $\mathcal{A}$.
  Send $(\textbf{secret share}, sid, sk_i)$ to A_i, $i = 1, \ldots, n$.
- *Upon receiving $(\textbf{message}, sid, m)$ from V_i store $(\textbf{message}, sid, (V_i, m))$ and send it to $\mathcal{A}$.*
- *Upon receiving $(\textbf{no-block}, sid, (V_i, m))$ from $\mathcal{A}$, check if some $(\textbf{message}, sid, (V_i, m))$ has been stored. In that case store $(\textbf{post}, sid, (V_i, m))$. Ignore subsequent* **no-block** *messages for V_i with* SID *sid from $\mathcal{A}$.*
- *Upon receiving $(\textbf{tally}, sid)$ from T send it to $\mathcal{A}$.*
  Ignore subsequent $(\textbf{tally}, sid)$ messages.
- *Upon receiving $(\textbf{no-block}, sid, \textbf{tally})$ send all stored $(\textbf{post}, sid, (V_i, m))$ messages as delayed output to $A_1, \ldots, A_n$.*
  Ignore subsequent $(\textbf{no-block}, sid, \textbf{tally})$ messages.
- *Upon receiving $(\textbf{post}, sid, m)$ from A_i send $(\textbf{post}, sid, (A_i, m))$ as delayed output to $A_1, \ldots, A_n$.*

Figure 18: Functionality $\mathcal{F}^{\text{mod}}_{\text{KM}}$

The modified Voting Protocol

We finally modify the $\mathcal{F}_{\text{KM}}$-hybrid protocol $\pi_{\text{Voting}}^{\mathcal{F}_{\text{KM}}}$ to obtain a $(\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}})$-hybrid protocol $\pi_{\text{Voting}}^{(\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}})}$ which instead of invoking $\mathcal{F}_{\text{KM}}$ now invokes the ideal functionalities $\mathcal{F}_{\text{DKG}}$, $\mathcal{F}_{\text{BB}}$, $\mathcal{F}_{\text{BC}}$ in the decomposition (A.1) of $\mathcal{F}_{\text{KM}}$.

Definition 41. *Assume we have given a Σ-protocol π_Σ, a homomorphic (t, n)-threshold cryptosystem (E, D) and a set of possible votes $\mathcal{V}$. The $(\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}})$-hybrid protocol $\pi_{\text{Voting}}^{(\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}})}$ is then defined as follows.*

Parties *: $V_1, \ldots, V_m$ (voters) and $A_1, \ldots, A_n$ (authorities) and T (tallying supervisor)*

Access to*: $\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}}$ and a random oracle $\mathcal{O}$*

(V1) Invoke $\mathcal{F}_{\text{DKG}}$ to establish the key generation and distribution. As a result, each party obtains the public key pk and each authority obtains the verification keys and its secret key.

(V2) Each V_i with pk on its `IC` *tape and a valid vote v_i on the* `input` *tape computes $x_i := E(pk, \text{Enc}(v_i))$, then chooses some a_i at random and queries $\mathcal{O}$ with (x_i, a_i, aux_i) to get a challenge e_i, where $aux_i := (pk, sid, V_i)$.*
Finally, he creates a proof $p_i = (a_i, z_i, aux_i)$ that $v_i \in \mathcal{V}$ using π_Σ.

V_i then sends $(\textbf{message}, sid, (x_i, p_i))$ to $\mathcal{F}_{\text{BB}}$.

(V3) When authority A_j has pk and the verification key vk_j on the `IC` *tape it does the following. When receiving a bunch of votes it extracts those votes $(v_{i_1}, \ldots, v_{i_\ell})$ with valid proofs. It then computes $C := \prod_{k=1}^{\ell} v_{i_k}$ and the corresponding decryption share ds_j. Finally, he chooses some a_j at random, queries $\mathcal{O}$ with (ds_j, a_j, aux_j) to get a challenge e_j, where $aux_j := (pk, sid, A_j)$ and creates a proof p_j using π_Σ and its verification key vk_j.*

A_j then sends $(\textbf{broadcast}, sid_{\text{loc}}, (\{A_1, \ldots, A_n\}, (\textbf{post}, sid, (ds_j, p_i))))$ to $\mathcal{F}_{\text{BC}}$.

(V4) Each authority A_j uses the first t decryption shares $ds_{j_1}, \ldots, ds_{j_t}$ of C with valid proofs to decrypt C to $\text{Enc}(\sigma) := D(pk, C)$.

When A_j has $(\textbf{tally}, sid)$ sent by T on its `input` *tape, then A_j outputs $(\textbf{result}, sid, \sigma)$.*

Figure 19: Protocol $\pi_{\text{Voting}}^{(\mathcal{F}_{\text{DKG}}, \mathcal{F}_{\text{BB}}, \mathcal{F}_{\text{BC}})}$

The modified Voting Functionality

Within the ideal voting functionality we do only have to add the tallying supervisor T which now sends the $(\textbf{result}, sid)$ message to $\mathcal{F}_{\text{Voting}}$ when it is instructed to do so by the environment.

Definition 42. *The modified ideal voting functionality* $\mathcal{F}_{\text{VOTING}}^{\text{mod}}$ *is defined as follows.*

Parties *: adversary* $\mathcal{S}$*,* $V_1, \ldots, V_m$ *(voters),* $A_1, \ldots, A_n$ *(authorities) and* T *(tallying supervisor)*

- *Upon receiving* $(\textbf{vote}, sid, (V_i, v))$ *from* V_i *store it and send* $(\textbf{vote}, sid, V_i)$ *to* $\mathcal{S}$*. Ignore subsequent* **vote** *messages from* V_i *with* SID *sid.*
- *Upon receiving* $(\textbf{no-block}, sid, V_i)$ *from* $\mathcal{S}$*, check if some* $(\textbf{vote}, sid, (V_i, v))$ *has been stored. In that case add* v *to the result* σ*. Ignore subsequent* **no-block** *messages for* V_i *with* SID *sid from* $\mathcal{S}$*.*
- *Upon receiving* $(\textbf{result}, sid)$ *from* T *compute the final result* σ_{FINAL}*, store it and send* $(\textbf{result}, sid, \sigma_{\text{FINAL}})$ *to* $\mathcal{S}$*.*
- *Upon receiving* $(\textbf{deliver-result}, sid, A_j)$ *from* $\mathcal{S}$*, send* $(\textbf{result}, sid, \sigma_{\text{FINAL}})$ *to* A_j*.*

Figure 20: Functionality $\mathcal{F}_{\text{VOTING}}^{\text{mod}}$

A.2. Representation of Quadratic Residue

The following lemma is due to the explanations given in [NS11, Appendix F].

Lemma 9. *Let* $N = pq$*. Let* g_{p^2} *and* g_{q^2} *be fixed generators of* Q_{p^2} *and* Q_{q^2}*, respectively. Then every* $v \in Q_{N^2}$ *has a representation* (v_p, v_q) *with* $v_p = (v_{p,1}, \ldots, v_{p,j_p})$ *and* $v_q = (v_{q,1}, \ldots, v_{q,j_q})$*, where each* $v_{p,j}$ *or* $v_{q,j}$ *corresponds to exactly one prime number in the factorization of the order of* $g_{p^2} \in Q_{p^2}$ *or* $g_{q^2} \in Q_{q^2}$*, respectively.*

Proof. Since $Q_{N^2} \cong Q_{p^2} \times Q_{q^2}$ by the Chinese remainder theorem, we can represent any $v \in Q_{N^2}$ by two integers $v_p \in \mathbb{Z}_{|Q_{p^2}|}$ and $v_q \in \mathbb{Z}_{|Q_{q^2}|}$ such that

$$\begin{aligned} v &\equiv g_{p^2}^{u_p} \mod p^2, \\ v &\equiv g_{q^2}^{u_q} \mod q^2. \end{aligned}$$

Since $|Q_{p^2}| = p(p-1)/2$ and $|Q_{q^2}| = q(q-1)/2$, these are the orders of the elements g_{p^2} and g_{q^2}, respectively. If we decompose these orders into

$$\frac{p(p-1)}{2} = \prod_{j=1}^{j_p} p_j^{\nu_j} \text{ and } \frac{q(q-1)}{2} = \prod_{j=1}^{j_q} q_j^{\nu'_j}$$

we can represent $v_p \in \mathbb{Z}_{\frac{p(p-1)}{2}}$ and $v_q \in \mathbb{Z}_{\frac{q(q-1)}{2}}$ as tuples

$$v_p = (v_{p,1}, \ldots, v_{p,j_p}) \text{ and } v_q = (v_{q,1}, \ldots, v_{q,j_q})$$

uniquely by choosing generators $v_{p,j} \in \mathbb{Z}_{p_j^{\nu_j}}$ and $v_{q,j} \in \mathbb{Z}_{q_j^{\nu'_j}}$, respectively, by the Chinese remainder theorem. □

List of Figures

Bibliography

[AF04] Masayuki Abe and Serge Fehr, *Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography*, Advances in Cryptology – CRYPTO 2004 (Matt Franklin, ed.), Lecture Notes in Computer Science, vol. 3152, Springer Berlin / Heidelberg, 2004, DOI 10.1007/978-3-540-28628-8_20, pp. 317–334.

[BF88] M. Blum and P. Feldman, *Non-interactive zero-knowledge and its applications (extended abstract)*, STOC '88: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pp. 103–112.

[BF97] Dan Boneh and Matthew K. Franklin, *Efficient generation of shared RSA keys (extended abstract)*, Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Springer London, 1997, pp. 425–439.

[BFP+01] Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Jacques Stern, and Guillaume Poupard, *Practical multi-candidate election system*, Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing (New York, NY, USA), PODC '01, ACM, 2001, pp. 274–283.

[BG93] Mihir Bellare and Oded Goldreich, *On defining proofs of knowledge*, Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '92, Springer London, 1993, pp. 390–420.

[BT94] Josh Benaloh and Dwight Tuinstra, *Receipt-free secret-ballot elections (extended abstract)*, Proceedings of the 26th Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '94, ACM, 1994, pp. 544–553.

[Can01] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols*, Proceedings of the 42nd IEEE symposium on Foundations of Computer Science (Washington DC, USA), FOCS '01, IEEE Computer Society, 2001, Full paper at `http://eprint.iacr.org/2000/067`, pp. 136–145.

[CCM08] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers, *Civitas: Toward a secure voting system*, IEEE Symposium on Security and Privacy (Los Alamitos, CA, USA), IEEE Computer Society, 2008, Full paper at `http://hdl.handle.net/1813/7875`, pp. 354–368.

[CDNO96] Ran Canetti Cynthia, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky, *Deniable encryption*, In Crypto '97, Springer Berlin / Heidelberg, 1996, pp. 90–104.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers, *Proofs of partial knowledge and simplified design of witness hiding protocols*, Advances in Cryp-

tology — CRYPTO '94 (Yvo Desmedt, ed.), Lecture Notes in Computer Science, vol. 839, Springer Berlin / Heidelberg, 1994, DOI 10.1007/3-540-48658-5_19, pp. 174–187.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers, *A secure and optimally efficient multi-authority election scheme*, Advances in Cryptology — EUROCRYPT '97 (Walter Fumy, ed.), Lecture Notes in Computer Science, vol. 1233, Springer Berlin / Heidelberg, 1997, DOI 10.1007/3-540-69053-0_9, pp. 103–118.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai, *Universally composable two-party and multi-party secure computation*, Proceedings of the 34th annual ACM symposium on Theory of Computing, STOC '02, ACM, New York, NY, USA, 2002, pp. 494–503.

[DF11] Markus Dürmuth and David Mandell Freeman, *Deniable encryption with negligible detection probability: an interactive construction*, Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology, EUROCRYPT'11, Springer Berlin / Heidelberg, 2011, pp. 610–626.

[DGS02] Ivan Damgård, Jens Groth, and Gorm Salomonsen, *The theory and implementation of an electronic voting system*, Kluwer Academic Publishers, 2002, pp. 1–26.

[DJ01] Ivan Damgård and Mats Jurik, *A generalisation, a simplification and some applications of paillier's probabilistic public-key system*, Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography, PKC '01, Springer London, 2001, pp. 119–136.

[DK01] Ivan Damgård and Maciej Koprowski, *Practical threshold RSA signatures without a trusted dealer*, Advances in Cryptology — EUROCRYPT 2001 (Birgit Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer Berlin / Heidelberg, 2001, 10.1007/3-540-44987-6_10, pp. 152–165.

[DKR06] Stephanie Delaune, Steve Kremer, and Mark Ryan, *Coercion-resistance and receipt-freeness in electronic voting*, Proceedings of the 19th IEEE workshop on Computer Security Foundations (Washington DC, USA), IEEE Computer Society, 2006, pp. 28–42.

[DN03] Ivan Damgård and Jesper Buus Nielsen, *Universally composable efficient multiparty computation from threshold homomorphic encryption*, CRYPTO), 2003, pp. 247–264.

[Fel87] Paul Feldman, *A practical scheme for non-interactive verifiable secret sharing*, Proceedings of the 28th Annual Symposium on Foundations of Computer

Science (Washington DC, USA), SFCS '87, IEEE Computer Society, 1987, pp. 427–438.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung, *Robust efficient distributed rsa-key generation*, Proceedings of the 30th Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 663–672.

[FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta, *A practical secret voting scheme for large scale elections*, Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, ASIACRYPT '92, Springer London, 1993, pp. 244–251.

[FPS01] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern, *Sharing decryption in the context of voting or lotteries*, Proceedings of the 4th International Conference on Financial Cryptography, FC '00, Springer London, 2001, pp. 90–104.

[FS01] Pierre-Alain Fouque and Jacques Stern, *Fully distributed threshold RSA under standard assumptions*, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT '01, Springer London, 2001, pp. 310–330.

[GGR09] Ryan W. Gardner, Sujata Garera, and Aviel D. Rubin, *Coercion resistant end-to-end voting*, Springer Berlin / Heidelberg, 2009, pp. 344–361.

[GL05] Shafi Goldwasser and Yehuda Lindell, *Secure multi-party computation without agreement*, vol. 18, Springer New York, 2005, 10.1007/s00145-005-0319-z, pp. 247–287.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game*, Proceedings of the 19th Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '87, ACM, 1987, pp. 218–229.

[Gro04] Jens Groth, *Evaluating security of voting schemes in the universal composability framework*, Applied Cryptography and Network Security (Markus Jakobsson, Moti Yung, and Jianying Zhou, eds.), Lecture Notes in Computer Science, vol. 3089, Springer Berlin / Heidelberg, 2004, DOI 10.1007/978-3-540-24852-1_4, pp. 46–60.

[GS08] Jens Groth and Amit Sahai, *Efficient non-interactive proof systems for bilinear groups*, Advances in Cryptology – EUROCRYPT 2008 (Nigel Smart, ed.), Lecture Notes in Computer Science, vol. 4965, Springer Berlin / Heidelberg, 2008, DOI 10.1007/978-3-540-78967-3_24, pp. 415–432.

[HB09] Jaydeep Howlader and Saikat Basu, *Sender-side public key deniable encryption scheme*, IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 9–13.

[HNBM11] Jaydeep Howlader, Vivek Nair, Saikat Basu, and A. K. Mal, *Uncoercibility in e-voting and eauctioning mechanisms using deniable encryption*, vol. 3(2), 2011, pp. 97–109.

[Ibr09] Maged Hamada Ibrahim, *A method for obtaining deniable public-key encryption*, International Journal of Network Security (IJNS), vol. 8, 2009, pp. 159–165.

[JCJ05] A. Juels, D. Catalano, and M. Jakobsson, *Coercion-resistant electronic elections*, Proceedings of the 2005 ACM workshop on Privacy in the electronic society (New York, NY, USA), WPES '05, ACM, 2005, pp. 61–70.

[KKK08] Marek Klonowski, Przemyslaw Kubiak, and Miroslaw Kutylowski, *Practical deniable encryption*, SOFSEM 2008: Theory and Practice of Computer Science, Lecture Notes in Computer Science, vol. 4910, Springer Berlin / Heidelberg, 2008, 10.1007/978-3-540-77566-9_52, pp. 599–609.

[Lin03] Y. Lindell, *Parallel coin-tossing and constant-round secure two-party computation*, vol. 16, Springer New York, 2003, DOI 10.1007/s00145-002-0143-7, pp. 143–184.

[Lin11] Yehuda Lindell, *Highly-efficient universally-composable commitments based on the ddh assumption*, Advances in Cryptology – EUROCRYPT 2011 (Kenneth Paterson, ed.), Lecture Notes in Computer Science, vol. 6632, Springer Berlin / Heidelberg, 2011, DOI 10.1007/978-3-642-20465-4_25, pp. 446–466.

[Lip05] Helger Lipmaa, *Secure electronic voting protocols*, The Handbook of Information Security, vol. 2, John Wiley & Sons, Inc., 2005.

[NS11] Takashi Nishide and Kouichi Sakurai, *Distributed paillier cryptosystem without trusted dealer*, Information Security Applications (Yongwha Chung and Moti Yung, eds.), Lecture Notes in Computer Science, vol. 6513, Springer Berlin / Heidelberg, 2011, Full version at `http://itslab.csce.kyushu-u.ac.jp/~nishide/nishide-DistPai.pdf`, pp. 44–60.

[PAB+05] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee, *Multiplicative homomorphic e-voting*, Progress in Cryptology - INDOCRYPT 2004 (Anne Canteaut and Kapaleeswaran Viswanathan, eds.), Lecture Notes in Computer Science, vol. 3348, Springer Berlin / Heidelberg, 2005, DOI 10.1007/978-3-540-30556-9_6, pp. 1403–1418.

[Pai99] Pascal Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, Advances in Cryptology — EUROCRYPT '99 (Jacques Stern, ed.), Lecture Notes in Computer Science, vol. 1592, Springer Berlin / Heidelberg, 1999, 10.1007/3-540-48910-X_16, pp. 223–238.

[PB11] Kun Peng and Feng Bao, *Efficient multiplicative homomorphic e-voting*, Information Security (Mike Burmester, Gene Tsudik, Spyros Magliveras, and Ivana Ilic, eds.), Lecture Notes in Computer Science, vol. 6531, Springer Berlin / Heidelberg, 2011, DOI 10.1007/978-3-642-18178-8_32, pp. 381–393.

[Ped91] Torben Pedersen, *A threshold cryptosystem without a trusted party*, Advances in Cryptology — EUROCRYPT '91 (Donald Davies, ed.), Lecture Notes in Computer Science, vol. 547, Springer Berlin / Heidelberg, 1991, DOI 10.1007/3-540-46416-6_47, pp. 522–526.

[Pie07] W. Pieters, *La volonté machinale*, Ph.D. thesis, Institute for Programming research and Algorithmics (IPA), 2007.

[Sha79] Adi Shamir, *How to share a secret*, vol. 22, ACM, November 1979, pp. 612–613.

[Sho00] Victor Shoup, *Practical threshold signatures*, Proceedings of the 19th international conference on Theory and application of cryptographic techniques, EUROCRYPT'00, Springer Berlin / Heidelberg, 2000, pp. 207–220.

[SMC95] József Sándor, Dragoslav Mitrinovic, and Borislav Crstici, *Handbook of number theory*, vol. 1, Springer Netherlands, 1995.

[UMQ09] D. Unruh and J. Müller-Quade, *Universally composable incoercibility*, Full version. Preprint on IACR ePrint 2009/520, October 2009.

[UMQ10] ———, *Universally composable incoercibility*, Advances in Cryptology – CRYPTO 2010, Lecture Notes in Computer Science, vol. 6223, Springer Berlin / Heidelberg, 2010, DOI 10.1007/978-3-642-14623-7_22, pp. 411–428.

[Unr07] Dominique Unruh, *Protokollkomposition und komplexität*, Ph.D. thesis, Universität Karlsruhe (TH), 2007, In German.

[Wik05] Douglas Wikström, *Universally composable dkg with linear number of exponentiations*, Security in Communication Networks (Carlo Blundo and Stelvio Cimato, eds.), Lecture Notes in Computer Science, vol. 3352, Springer Berlin / Heidelberg, 2005, 10.1007/978-3-540-30598-9_19, pp. 263–277.

Selbstständigkeitserklärung

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 29.11.2011