NTNU
Innovation and Creativity

Electronic voting systems

Rune Steinsmo Ødegård

Master of Science in Physics and Mathematics
Submission date: October 2006
Supervisor: Aslak Bakke Buan, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

The main aim of the thesis is to understand and compare different mathematical models for electronic voting systems.

Currently there are several competing models, two of them are:

1. Systems based on homomorphic encryption
2. Systems based on verifiable secret sharing

In the thesis, the mathematical background for these systems will be studied, with particular emphasis on the security issues of the relevant sub-protocols. Furthermore, it is of interest to compare security issues as well as efficiency issues of the two different models.

Assignment given: 12. June 2006
Supervisor: Aslak Bakke Buan, MATH

Electronic Voting Systems

Rune Steinsmo Ødegård

October 29, 2006

Foreword

This document is the Master thesis of Rune Steinsmo Ødegård, prepared in the fall of 2006, as the final part of my Master of Science degree from the Norwegian University of Science and Technology.

I would like to thank my supervisor Aslak Bakke Buan for an interesting and intriguing subject, and for the guidance given during the preparation and writing of this thesis. Furthermore, I would like to thank fellow student and LaTeX-mentor Sigurd Segtnan for proofreading the final document, and my mother Unni Steinsmo for general tips on the subject of writing scientific articles.

Trondheim, October 29, 2006

Abstract

We present the cryptographic primitives needed in the construction of electronic voting systems based on homomorphic encryptions and on verifiable secret sharing. Then "The theory and implementation of an electronic voting system" by Ivan Damgård, Jens Groth and Gorm Salomonsen is presented as an example of electronic voting systems based on homomorphic encryptions, while "Multi-authority secret-ballot election with linear work" by Ronald Cramer, Matthew Franklin, Berry Schoenmakers and Moti Yung is presented as an example of electronic voting systems based on verifiable secret sharing. Moreover, the mathematical background for these systems are studied with particular emphasis on the security issues of the relevant sub-protocols.

Comparing these two examples we find that the presented voting system based on verifiable secret sharing is more secure then the one based on homomorphic encryptions, both in regard to privacy and robustness. On the other hand, we find that the presented voting system based on homomorphic encryptions is more efficient then the one based on verifiable secret sharing.

Contents

1

Introduction

An electronic voting system is viewed as a set of protocols that allow a collection of voters to cast their votes, while enabling a collection of authorities to collect the votes, compute the final tally, and communicate the final tally that is checked by the talliers [CFSY96]. In the cryptographic literature on voting systems, three important requirements are identified:

- **Universal verifiability:** After the election any party, including a passive observer, can convince himself that the election was fair in the sence that the published final tally is computed correctly from ballots that were correctly cast.
- **Privacy:** All individual votes will be kept secret from any (reasonably sized) coalition of parties. Possible degrees of privacy is depicted in Figure 1.1 from [Saf01].
- **Robustness:** The results reflect all submitted and well-formed ballots correctly, even if some voters and/or possibly some entities running the election cheat. The possible degrees of robustnuss are the same as the ones of privacy.

In addition *no vote-duplication* and *receipt-freeness* are usually considered desirable properties of the voting system. No vote-duplication means that is should be impossible to copy another voters vote (even without knowing what the copied vote is). While receipt-freeness means that the voter should not be able to get a receipt from the voting system which shows which way the voter voted[1]. This property is desirable since the receipt can be used for vote buying and coercion, which is why the property sometimes is referred to *non-coercibility*.

Various fundamentally different approaches are known in the literature. One of these is to use *blind signatures and and anonymous channels* [FOO93], where the channels usually are implemented using *MIX nets* [Nef01]. The idea in such a voting system is that the voter prepares a ballot in clear-text. He then interacts with an authority to show that he is eligible to vote and

[1]The notion of receipt for electronic voting systems was introduced by Benaloh and Tuinstra in [BT94] where they showed how previous election protocols all suffer from this defect.

Privacy strength	**Relies upon**	**Attacks**
Policy	correctly following a communication protocol defined by policy	1. circumvent policy 2. collusion 3. court order
Computational	mathematical expressions and their correct implementation in hardware/software	1. break the code 2. obtain the keys 3. collusion 4. court order
Information-theoretic	unprovable (non-computable) relations	1. obtain secrets 2. collusion 3. court order
Fails safe	unknowable relations	none

Figure 1.1: Four cases of privacy ranked in increasing degree of privacy strength.

has not already voted. If that is the case the authority issues a blind signature on the ballot. Informally this means that the voter obtains the authorities digital signature on the ballot, without the authority learning anything about the content of the ballot. And, on the other hand, a voter can not obtain the signature without interacting with the authority. The voter then sends the ballot to another authority which is responsible for tallying the votes. In order to maintain privacy this must be done using anonymous channels, which cryptographically can be implemented using MIX-nets, or the channel may rely on some physical assumption. After all ballots have been received they can be counted directly ignoring the ballots without the relevant authority's signature [DGS03].

Another approach is to use several servers to count the votes and have voters *verifiable secret share* the votes among the servers [CFSY96, Sch99]. In such a voting system each voter sends a share of his ballot to each authority. This is done in such a way that fewer then some threshold t gets no information on the vote, while t or more authorities can go together and reconstruct the ballot. In addition the voter must convince the authorities that the ballot was correctly constructed, and so he is prevented from voting twice or voting incorrectly. When all the votes have been cast the servers can jointly compute the result of the election without any side information becoming public.

A final approach is to use *homomorphic encryptions* [DGS03, CGS97, BFP$^+$01, DJN03]. In such a voting system the voter publishes an encryption of his vote represented by a number, which is encrypted using a homomorphic public-key cryptosystem. When submitting his vote, the voter must identify himself to prove that he is eligible to vote, and has not voted before. In addition he must prove knowledge of the fact that his encryption contains a valid vote. This does not violate privacy since all individual votes remain encrypted, and the proof is *zero-knowledge.* Informally this means that the proof is done in such a way that the authorities gain no information about the content of the encryption, except the fact that it contains a valid vote. Using the homomorphic property of the cryptosystem the authorities can compute an encryption to the final tally from the encrypted votes by simple binary operations (usually multiplication). All that is left is to decrypt the final results. This can be done securely assuming the private-key of the cryptosystem has been secretly shared among a set of authorities. The shares have to be constructed in such a way that fewer than some threshold t of the authorities get no information about the key, while a set of t or more authorities can go together and jointly decrypt the results. This is often called *threshold decryption.*

Our goal in this thesis is to understand and compare voting systems based on verifiable secret sharing and voting systems based on homomorphic encryptions. For this purpose we will use "Multi-authority secret-ballot election with linear work" by Ronald Cramer, Matthew Franklin, Berry Schoenmakers and Moti Yung [CFSY96] as an example of electronic voting systems based on verifiable secret sharing, while "The theory and implementation of an electronic voting system" by Ivan Damgård, Jens Groth and Gorm Salomonsen [DGS03] will be used as an example of electronic voting systems based on homomorphic encryptions. The mathematical background for these systems will be studied, with particular emphasis on the security issues of the relevant subprotocols. Furthermore, it is of interest to compare security issues as well as efficiency issues of the two different voting systems.

The disposition of the thesis is as follows: In Chapter 2 we define the cryptographic primitives we need for our voting systems. In Chapter 3 we present the voting system based on homomorphic encryption, while the voting system based on verifiable secret sharing is presented in Chapter 4. The security and efficiency issues of the two presented systems are compared in Chapter 5, while Chapter 6 concludes the paper.

2

Cryptographic primitives

2.1 Turing machines

Turing machines[1] can be seen as an abstraction of computers as we know them today. We will give an informal description based on the description found in [Cra96], while a formal definition can be found in for instance [LP98]. To perform some computational task, the *Turing machine* is provided with an adequate list of elementary instructions (an algorithm). Similar to a programming language, each of these instructions must be chosen from some finite set of eligible instructions. Any (private) input to the machine is placed on the part of memory called knowledge tape. The machine has also access to an auxiliary tape which can be used to read and write intermediate results. When the computations dictated by the algorithm have been completed, the machine halts and output the outcome on the output tape. A Turing machine is called *probabilistic* if it also has access to a random tape, which is part of the memory allocated for storage of random bits. In this case the machine is allowed to read random bits and use them in any of its computations. A Turing machine is called *polynomial time*, or sometimes *efficient*, if the total number of read and write operations to and from the tape is bounded by some polynomial in the length of the input. Any computation which can not be performed by a polynomial time Turing machine is called *infeasible* or *intractable*. The probability, $\epsilon(\cdot)$, that a polynomial time Turing machine can solve some computational problem is said to be *negligible* in some security parameter l if for any polynomial $p(\cdot)$, $\epsilon(l) \leq 1/p(l)$ for all large enough l.

We say that an interactive Turing machine is given to us as a *black-box* if we are allowed to run the machine as many times as we desire (but at most a polynomial number of times). Moreover, we are allowed to put any strings of our choice on its communication tape. If we in addition are allowed to supply the random bits for the random tape we say that the machine is given to us as a *rewindable* black-box.

Before we continue a word of warning about the definition of intractable, infeasible and negligible. They are based on the asymptotic behavior of the adversary as we increase the value of the security parameter. This is mathematically convenient when doing proofs, and has nice connections to standard complexity theory, but one should take care when evaluating the meaning in practice. The reason for this is that the definition implicitly classifies everything that can be solved in polynomial time as "easy" and anything else as being "hard". This is not always accurate in real life. Even if a problem is asymptotically hard, it might still be easy for those input sizes we want in a particular application. This does not mean that asymptotical security results are worthless, only that usage of a scheme in real life should always be supplemented with an analysis of practical state of the art solutions to the (supposedly) hard problem we base ourselves on [Dam99].

2.2 Intractability assumptions

As most other cryptographic protocols, such as for instance the RSA public-key cryptosystem, we will base the construction of our voting systems on the assumption that some computational problem is intractable. In the remainder of this thesis we will assume that the following three problems are intractable.

[1] Named after their inventor Alan Turing, who is often considered to be the father of modern computer science.

Definition. ([Sti02]) *The discrete logarithm problem:* Given a group G, an element $g \in G$ of order n, and an element $h \in \langle g \rangle$, find the unique integer a, $0 \leq a \leq n-1$ such that $g^a = h$.

Definition. ([DGS03]) *The generalized Decision Diffie Hellman problem:* Given a ring R an element $g \in R$ of order n and three elements $g^a, g^b, g^c \in \langle g \rangle$, determine if $c = ab \bmod n$.

Definition. ([RK03]) *The RSA problem:* Given a RSA public-key (n, e) and a ciphertext y, find m such that $m^e \bmod n = y$.

We will refer to the assumption that these problems are intractable as the discrete logarithm assumption, the generalized DDH assumption and the RSA assumption respectively.

Note that the generalized DDH assumption is as least as strong as the discrete logarithm assumption. Given an oracle $\mathrm{DL}(\cdot)$, that on input g^a outputs a. We can solve the DDH problem on input (g^a, g^b, g^c), by checking if $\mathrm{DL}(g^a) \cdot \mathrm{DL}(g^b) \bmod n = \mathrm{DL}(g^c)$.

2.3 Interactive proof systems

An interactive proof system, formally defined below, is a form of conversation between a prover and a verifier, where the prover tries to convince the verifier that a certain (usually mathematical) statement is true. If the prover is honest then he will always succeed, while if the prover is dishonest he will almost certainly fail. An interactive proof system is said to be zero-knowledge if all the verifier learns from talking to an honest prover is the truthfullnes of the statement, and nothing more[2].

To ensure privacy we want the voter to be able to prove the correctness of an encrypted vote using such a zero-knowledge proof system. Unfortunately, zero-knowledge proofs usually require a large number of repetitions before the desired level of confidence is reached, which would be too time-consuming for a bigscale election. Moreover, zero-knowledge is not necessarily preserved under general composition of protocols [FS90]. Therefore we will instead look into a class of honest-verifier-zero-knowledge proofs called Σ-protocols. First we need the following definitions from [Dam05].

Definition. Let $R = \{(x, w)\} \subset \{0,1\}^* \times \{0,1\}^*$ be a binary relation where membership can be tested in polynomial time. For some $(x, w) \in R$, we may think of x as an instance of some computational problem, and w as the solution to that instance. We call w a *witness* for x, and for any x, its *witness set* $w(x)$ is the set of w's such that $(x, w) \in R$.

Definition. Let κ be a function from bit strings to the interval $[0, \ldots, 1]$. The protocol (P, V) is said to be a proof of knowledge for the relation R with knowledge error κ, if the following is satisfied:

- *Completeness:* On common input x, if the honest prover P gets as private input w such that $(x, w) \in R$, then the verifier V always accepts.

- *Knowledge soundness:* There exists a probabilistic algorithm M called the knowledge extractor. This M gets input x and rewindable black-box access to the prover and attempts to compute w such that $(x, w) \in R$. We require that the following holds: For any prover P^*, let $\epsilon(x)$ be the probability that V accepts on input x. There exists a constant c such that whenever $\epsilon(x) > \kappa(x)$, M will output a correct w in expected time at most

$$\frac{|x|^c}{\epsilon(x) - \kappa(x)}$$

where access to P^* counts as one step only, and $|x|$ is the length of the bit-string x.

[2]Zero-knowledge was first introduced by Goldwasser et. al. in [GMR85]. We refer the interested reader to [Dam99] for a formal definition of (and a good introduction to) the concept.

One may think of the error $\kappa(x)$ as the probability that the prover can on input x convince the verifier without knowing a correct w. The knowledge soundness requirement assures that being better than $\kappa(x)$ requires some ability to compute w. Moreover, the requirement also assures that the provers ability to compute w gets more efficient the better the prover is at convincing the verifier. This is based on the paradigm that a machine "knows" something if it can be used to compute it efficiently. That is, if the simulator M can pull the information out of P^*, it must somehow have been in there [Dam05].

2.3.1 Σ-protocols

We will be concerned with protocols of the following form, where x is a common input to both the prover (P) and verifier (V), while the prover has a private input w such that $(x, w) \in R$:

1. P sends a message a
2. V sends a random t-bit string e (we may also say that e has challenge length t).
3. P sends a reply z, and V decides to accept or reject based on the data (x, a, e, z) he has seen.

We will assume throughout that both P and V are probabilistic polynomial time Turing machines, so that P's only advantage over V is that he knows w.

Definition. ([Dam05]) A protocol $\mathcal{P}$ is said to be a Σ-protocol[3] for relation R if the protocol is on the above 3-move form and satisfies the following criteria:

- *Completeness:* If P and V follow the protocol the verifier always accepts.
- *Special soundness:* From any x and any pair of accepting conversations (a, e, z), (a, e', z') on input x, where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$.
- *Special honest-verifier zero-knowledge:* There exists a polynomial-time simulator M, which on input x and a random e outputs an accepting conversation of the form (a, e, z), with the same probability distribution as conversations between honest P and V on input x.

The following theorem from [Dam05] shows the link between Σ-protocols and proofs of knowledge. Moreover, the theorem shows that the probability that a prover in a Σ-protocol can convince verifier without knowing a correct witness is negligible in the challenge length t.

Theorem 2.1. *Let $\mathcal{P}$ be a Σ-protocol for relation R with challenge length t. Then $\mathcal{P}$ is a proof of knowledge with knowledge error 2^{-t}.*

Proof. Completeness is clear by definition.

The proof of special soundness (of Σ-protocols) being equivalent to knowledge soundness (of proofs of knowledge) with knowledge error 2^{-t} is beyond the scope of this thesis. We refer the interested reader to [Dam05]. □

Example. As a concrete example of a simple Σ-protocol we present Schnorr's protocol from [Sch91] for proving knowledge of a discrete log in group G of prime order q. The protocol is a proof of knowledge for the relation $R = \{(x, w)\} = \{((h, g, G), w) \mid h = g^w \text{ in } G\}$, and works as follows:

1. The prover chooses r at random in $\mathbb{Z}_q$, and sends $a = g^r$ to V.
2. The verifier chooses e at random in $\mathbb{Z}_{2^t}$ and sends it to P. Here t is fixed such that $2^t < q$.

[3]The idea of Σ-protocols as an abstract concept was first introduced by Cramer in his PhD-thesis [Cra96]. Spelled out, the "sig" part of sigma refers to "zig-zag" symbolizing the three moves, while the "ma" part is an abbreviation of "Merlin-Arthur." A proof system is said to be of the type Arthur-Merlin if verifier is expected to send only uniformly chosen bits (see [BM88]).

3. P sends $z = (r + ew) \bmod q$ to V, and V accepts if and only if $g^z = ah^e$.

Since $g^z = g^{r+ew} = g^r(g^w)^e = ah^c$, completeness trivially holds with probability 1. As for special soundness note that correct answers to two different challenges e_1, e_2, gives the two equations $z_1 = r + e_1w \bmod q$ and $z_2 = r + e_2w \bmod q$, which gives $w = (z_1 - z_2)(e_1 - e_2)^{-1} \bmod q$. So a cheating prover who does not know w can only be able to answer at most one challenge value correctly, since otherwise the above equation would imply that he was in fact able to compute w after all. Thus the error probability for this proof is 2^{-t}, as expected by Theorem 2.1 .

To simulate an accepting conversation simply choose at random $z \in G$ and $e \in \mathbb{Z}_q$ and compute $a = g^z h^{-e}$. Then clearly (a, e, z) has exactly the same probability distribution as real conversations between the honest prover and the honest verifier.

When the prover chooses $r \in \mathbb{Z}_q$ in the example above, he knows that he will at a later time reveal $z = r + ew \bmod q$. To preserve the zero-knowledge property, it is therefore important that the prover chooses r such that revealing z does not give away any information about w. This concept is called *shadowing*[4].

Definition. ([DGS03]) An element r in a ring R is said to be a *shadow* of $ew \in R$ if revealing $r + ew$ does not give away any information about w. If in addition r is chosen such that we cannot distinguish it from a properly chosen random element of R we call r a *random shadow*. We can speak of computational, statistical and perfect shadowing depending on how the shadow hides the underlying elements.

Trivially, by the special honest verifier zero-knowledge property, the verifier can not use the conversation with the prover to find the witness the prover is using. But if the prover knows a set of witnesses $w(x)$, can multiple executions of the protocol where the prover uses a different $w \in w(x)$ help the verifier guess what $w \in w(x)$ the prover is using. To explore this possibility we need the following definition based on a formal one from [FS90]

Definition. Let $\mathcal{P} = (P, V)$ be a proof of knowledge for relation $R = \{(x, w)\}$. If V cannot distinguish between two executions of the protocol which differ in the specific witness, $w \in w(x)$, the prover is using, the protocol is *witness indistinguishable*.

Proposition 2.1. ([CDS94]) Let $\mathcal{P}$ be a Σ-protocol, then $\mathcal{P}$ is witness indistinguishable.

Proof. Let (a, e, z) be the triplet formed from the conversation between the prover and the verifier. By the special honest verifier property, the use of any witness w leads to the distribution produced by the simulator. This means that the distribution of z given any fixed a and e is independent of w. The proposition then follows by noting that in conversations with a general verifier, the distribution of a, and hence e, is independent of w. $\square$

We note that in many concrete cases, this proposition is not interesting because there is only one witness, in which case witness indistinguishability is trivial and can not imply anything.

The following theorem from [FS90] shows that we can compose a new Σ-protocol by using other Σ-protocols as sub-protocols while still maintaining witness indistinguishability.

Theorem 2.2. Witness indistinguishability is preserved under general composition of protocols. Specially witness indistinguishability is preserved under both sequential composition, in which the protocols are executed one after the other, and parallel composition, in which all protocols are of the same type and run on the same input, and for each j, steps j in all protocols are executed at the same time.

Proof. The proof of this theorem is beyond the scope of this thesis. We refer the interested reader to [FS90]. $\square$

[4]The definition of shadows and shadowing is based on the little information from [DGS03]. No other literature on the concept has been found.

2.3.2 The OR-proof

Assume we are given a Σ-protocol $\mathcal{P}$ for a relation R, where x_0, x_1 is common input to P, V and w is a private input to P such that $(x_b, w) \in R$ where $b \in \{0,1\}$. We want to produce a Σ-protocol $\mathcal{P}_{OR}$ that allows a prover to show that given the two inputs x_0, x_1, he knows w, such that either $(x_0, w) \in R$ or $(x_1, w) \in R$, but without revealing which is the case.

The idea is that we will ask the prover to complete two instances of the protocol, one with respect to x_0 and the other with respect to x_1. If we give the prover a little freedom in choosing the challenges to answer, he can complete both instances. For x_b he can do this for real, while for x_{1-b} he will have to fake it using the simulator M. More precisely, consider the following protocol $\mathcal{P}_{OR}$[5]:

1. P computes the first message a_b in $\mathcal{P}$ using x_b, w as input.

   P chooses e_{1-b} at random and runs the simulator M on input x_{1-b}, e_{1-b}, let $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output.

   P sends a_0, a_1 to V

2. V chooses a random t-bit string s and sends it to P.

3. P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b in $\mathcal{P}$ to challenge e_b using x_b, a_b, e_b, w as input. He sends e_0, z_0, e_1, z_1 to V.

   V decides to accept the proof if and only if $s = e_0 \oplus e_1$ and both (a_0, e_0, z_0) and (a_1, e_1, z_1) are accepting conversations.

Theorem 2.3. ([Dam05]) Let $R_{OR} = \{((x_0, x_1), w) | \ (x_0, w) \in R \text{ or } (x_1, w) \in R\}$. The protocol $\mathcal{P}_{OR}$ above is a Σ-protocol for R_{OR}. Moreover, for any verifier V^*, the probability distribution of conversations between P and V^*, where w is such that $(x_b, w) \in R$, is independent of b.

Proof. Completeness trivially holds since if the prover is honest (a_b, e_b, z_b) is an accepted conversation by the completeness property of $\mathcal{P}$, while $(a_{1-b}, e_{1-b}, z_{1-b})$ is an accepted conversation by the special honest verifier zero-knowledge property of $\mathcal{P}$.

To verify special soundness let two accepting conversations $((a_0, a_1), s, (e_0, e_1, z_0, z_1))$, $((a'_0, a'_1), s', (e'_0, e'_1, z'_0, z'_1))$ with $s \neq s'$ be given. Then clearly, for some $c = 0$ or 1, $e_c \neq e'_c$. This means we can compute a witness w from the conversations (a_c, e_c, z_c),(a'_c, e'_c, z'_c) by the special soundness property of $\mathcal{P}$.

To simulate an accepting conversation simply choose e_0, e_1 at random subject to $s = e_0 \oplus e_1$ and run M twice, once on input (x_0, e_0) and once on input (x_1, e_1). It follows that $\mathcal{P}_{OR}$ satisfies special honest verifier zero-knowledge.

Finally assume we are given an arbitrary verifier V^*. Then observe that the distribution of conversations beteem P and V^* can be specified as follows. They have the form $(a_0, a_1), s, (e_0, e_1, z_0, z_1)$, and from the honest verifier property of $\mathcal{P}$ we see that a_0, a_1 are distributed as an honest prover in $\mathcal{P}$ would choose them. Then s has whatever distribution V^* outputs given x_0, x_1, a_0, a_1. Furthermore, e_0, e_1 are chosen at random subject to $s = e_0 \oplus e_1$. Finally, z_0 has whatever distribution the honest prover in $\mathcal{P}$ outputs, given that the input was x_0 and the first part of the conversation was a_0, e_0. A similar conclusion holds for z_1. This trivially holds for z_b, and follows from honest verifier zero-knowledge property for z_{1-b}. This shows that the probability distribution of conversations between P and V^* does not depend on b. $\square$

We note that, by the last claim in the theorem above, the protocol $\mathcal{P}_{OR}$ is witness indistinguishable as expected by Proposition 2.1 and Theorem 2.2.

[5] The construction of this protocol and the following theorem is based on the more general results of [CDS94], where a protocol for proving that one knows the solution to some subset of n problem instances is presented.

2.3.3 Non-interactive Σ-protocols

In [FS87] Fiat and Shamir proposed a heuristic method for making Σ-protocols non-interactive. This was later formalized as the random oracle model by Bellare and Rogaway in [BR93]. The idea is as follows:

Definition. ([Dam05]) A *random oracle* is an entity that initially chooses a random function $R : \{0,1\}^l \rightarrow \{0,1\}^t$ for some l, t. Then any player can send any bit-string a of length l to the oracle, who will return $R(a)$. Since R was completely random, $R(a)$ is a uniformly chosen string of length t, and is independent of a.

Given such an oracle, a Σ-protocol can be made non-interactive by having the prover ask the oracle for a random challenge instead of the verifier. The prover sends a to the oracle, answers the random challenge $e = R(a)$ resulting in z, and then sends (a, z) to the verifier. The verifier then calls the oracle with a as input to get the value e, and proceeds to check the answer z as he normally would have done.

The question is, how does the random oracle model affect a cheating prover or a cheating verifier? Since a cheating prover cannot get an oracle response on a without calling the oracle, he has no information on e before he has sent a. In a sense this is almost the same as talking to the verifier; the only difference is that a cheating prover can call the oracle on different values until he gets a challenge he can answer. But if the number of possible challenges is exponentially large, and the prover only has polynomial time, this is not a feasible strategy.

As for the verifier the random oracle model forces a cheating verifier to be honest since the challenges will always be random and uniformally chosen. Since we assume that the Σ-protocols are honest-verifier-zero-knowledge, the Σ-protocol becomes zero-knowledge in the random oracle model. For a more formal definition of zero-knowledge in this model see [Dam05].

To make this knowledge practical in real life we can use a hash function as a heuristic random oracle [Dam05]. Given a secure one-way hash function $\mathcal{H}$, the prover can answer the challenge $e = \mathcal{H}(a)$ resulting in z, and send (a, z) to the verifier, who checks if $(a, \mathcal{H}(a), z)$ is an accepting conversation. Using this method in an election scheme would presumably result in less communication complexity and therefore improved efficiency of the scheme. However, recently Xiaoyun Wang et.al presented a method for finding collision in SHA-1 [WYY05], as well as other popular hash functions [WFLY04]. No literature has been found to support whether or not the gain in communication complexity is worth the extra computational complexity of a slower and more secure hash algorithm such as the pending SHA-256 [Lan06].

2.4 Verifiable secret sharing

Secret sharing schemes were introduced independently by Shamir [Sha79] and Blakley [Bla79] in 1979, and since then much work has been put into the investigation of such schemes. The concept is as follows: A *dealer* has some secret s which he distributes to n players by giving each player *share* in the secret. This is done is such a way that a coalition of the players larger than some threshold t can work together to reconstruct s, while any set of less then t players have no information about the secret $(1 \leq t \leq n)$[6]. This is called a (t, n)-threshold scheme. A secret sharing scheme is said to be a verifiable secret sharing scheme if it satisfies the following definition from [Cra99][7].

Definition. A secret sharing scheme is called a *verifiable secret sharing scheme* if the following is satisfied.

1. If the dealer is honest, then the distribution of a secret s always succeeds, and the corrupted players gain no information about s as a result of the distribution phase. At reconstruction, the honest players recover s. These properties hold regardless of the behavior of the corrupted players.

[6] See[Bei96] for a more formal definition of secret sharing.

[7] The definition is an informal one based on the definition found in [Gen96], but it is satisfactory for our purpose.

2. If the dealer is corrupt, then the following holds. Either the dealer is deemed corrupt by the honest players, and all of them abort the distribution phase. Otherwis, the distribution phase is accepted by the honest players and *some* value s is uniquely fixed by the information held by the honest players as a result of the distribution phase. In the reconstruction phase, the honest players recover the value s. These properties hold regardless of the behavior of the corrupted players.

The verifiable secret sharing schemes constitute a particular interesting class of secret sharing schemes as they allow each player to verify that his share is consistent with the other shares, and therefore forces the dealer and players to be honest. Such protocols is an important tool in secure multi-party computation. First introduced by Goldwasser et.al in [GMW87], multi-party computation is a method that allows for a group of people to jointly compute on an agreed function without revealing their private data. This concept has surprisingly many application (see [DA01, Gol97]) and, as we shall see, it can also be an important tool in the construction of electronic voting systems.

As an example of a secret sharing scheme we will now present Shamir's scheme [Sha79] which the verifiable secret sharing scheme we will use in Chapter 4 is based upon. The scheme is fairly intuitive, and is based on the following well known fact: Two points in the plane define a line, three points define a parabola, and more generally will t points in the plane define a polynomial of degree $t-1$ (see Theorem 2.4 below). Lets take the line as an example and assume we want to distribute $s \in F$ where F is a field. To do this we simple draw a random line trough $(0, s)$ and send a point on this line to each player. Now, every pair of players can go together and calculate the line we used to distribute s, and then find the secret where this line crosses the y-axes. But no single player P_j can find s, since for every $\tilde{s} \in F$ there exists a line that goes trough $(0, \tilde{s})$ and player P_j's point. In a similar way we can use a random polynomial of degree t that goes through $(0, s)$ to distribute the secret. Then any set of at least t players can find the polynomial that goes through s, by the following theorem from [Cra99].

Theorem 2.4. Let F be a field, and let S be a finite collection of indices such that $|S| = t$, and finally let $(p_i, q_i), i \in S$ be a collection of t points in the plane $F \times F$. Then there exists a unique polynomial $f(x) \in F[x]$ of degree less then t such that $f(p_i) = q_i$ for all $i \in S$.

Proof. We will give a proof by construction using a method called *Lagrange interpolation.* For each $i \in S$ let

$$f_{S,i}(x) = \prod_{j \in S \setminus \{i\}} \frac{x - p_j}{p_i - p_j}$$

Notice that $\deg(f_{S,i}) = t-1$, $f_{S,i}(p_j) = 0$ for all $j \neq i$, and that $f_{S,i}(p_i) = 1$ for all $i \in S$. This means that

$$f(x) = \sum_{i \in S} q_i f_{S,i}(x)$$

is exactly the polynomial we are looking for. Moreover, since $f(x)$ is the sum of polynomials of degree $t-1$, $f(x)$ must be a polynomial of degree $t-1$ or less. To prove uniqueness assume there exists a polynomial $f'(x) \neq f(x)$ of degree less then t such that $f'(p_i) = f(p_i)$ for all i. Then $g(x) = f(x) - f'(x)$ is a polynomial with at least t zeros (one for each p_i), while its degree is less then t. This is only possible if $g(x) = 0$, so $f'(x) = f(x)$ after all. $\square$

We will now present Shamir's scheme which is based on the idea above. Let F be a finite field where $|F| > n$, where n is the total number of players. Let $(P_1, \ldots, P_n)$ be a set of players where each player P_i is associated with a distinct element $i \in F \setminus \{0\}$. Finally let $s \in F$ be a secret that the dealer wishes to distribute. The following is then a secret sharing scheme for n players with threshold t.

Shamir's secret sharing scheme

Distribution. The dealer chooses a random $f(x) = \rho_{t-1}x^{t-1} + \cdots + \rho_1 x + \rho_0$ by setting $\rho_0 = s$ and the remaining ρ_i to random elements from F. The dealer then sends $s_i = f(i)$ on a private channel to player P_i, where s_i is player P_i's share in s.

Reconstruction. Using the formula for Lagrange interpolation from the proof above, and defining

$$\lambda_{S,i} = f_{S,i}(0) = \prod_{j \in S \setminus \{i\}} \frac{-j}{i-j},$$

we see that a set S of t or more players efficiently can reconstruct s using the following formula:

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

Note that privacy is preserved in the information theoretical sence by the following. Let $|S| = t - 1$, then Lagrange interpolation of the following t points

$$\{(0, s'), (i_1, s_{i_1}), \ldots, (i_{t-1}, s_{i_{t-1}})\}, i_j \in S$$

yield a unique polynomial $f_{s'}$ for each $s' \in F$. So from the joint view of the players in S, each secret is equally likely and hence the shares held by S give no information about the real secret s. It follows that any set of less then $t - 1$ players can not reconstruct the secret either, since a smaller set hold even less information.

2.5 Homomorphic encryptions

One of the key problems in constructions of electronic voting systems is ensuring universal verifiability while maintaining privacy. To obtain privacy the ballot of each individual must in some way be encrypted, and in order to tally the ballots they must at one time be decrypted. The problem lies in ensuring both that there is no link between the decrypted ballot and the voter who cast it *and* that a passive observer can verify that each vote has been counted correctly. To solve this problem homomorphic encryption schemes are at the heart of most electronic voting systems.

Definition. ([Lip01]) A public-key cryptosystem Λ is a triple $\Lambda = (Gen, E, D)$, where E and D is a family of encryption and decryption functions respectivly, and Gen is an efficient key generation algorithm which generates a key pk. With this key there is an associated message space $\mathcal{M}_{pk}$, randomizer space $\mathcal{R}_{pk}$, ciphertext space $\mathcal{C}_{pk}$, an encryption function $E_{pk} : \mathcal{M}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$ and a decryption function $D_{pk} : \mathcal{C}_{pk} \to \mathcal{M}_{pk}$. We will assume that all three spaces $(\mathcal{M}_{pk}, \mathcal{R}_{pk}, \mathcal{C}_{pk})$ are Abelian groups with $\mathcal{C}_{pk}$ written multiplicatively. We say that the public-key cryptosystem $\Lambda = (Gen, E, D)$ is *homomorphic* if $E_{pk}(m_1; r_1)E_{pk}(m_2; r_2) = E_{pk}(m_1 + m_2; r_1 + r_2)$. Some examples of homomorphic cryptosystems are the Paillier cryptosystem [Pai99] and the Damgård-Jurik cryptosystem [DJ01].

Example. We now present a short example of how a homomorphic encryption scheme *can* be used to ensure universal verifiability in an electronic voting scheme. Assume we want to perform a yes/no election, and that we are given a homomorphic encryption scheme where a key pk already has been generated. To vote each participants simply encrypts -1 or 1 for no or yes respectively. That is $y_i = E_{pk}(b_i; r_i)$, where $b_i \in \{-1, 1\}$ and $r_i \in \mathcal{R}_{pk}$, is a vote by participant i on b_i. Let there be n participants, then decrypting $y = y_1 \cdots y_n$ would yield the result of the election, since

$$y = \prod_{i=1}^{n} y_i = E_{pk}(b_1 + \cdots + b_n; r) = E_{pk}(m; r),$$

where the first equality follows by definition, $r = \sum_{i=1}^{n} r_i$, and $m = \sum_{i=1}^{n} b_i$ represents the total number of yes votes minus the total number of no votes. Moreover, a passive observer can easily verify the results by doing the above calculations by himself.

Note that the above protocol will only work as desired if all participant are honest, that is if they are all following the protocol. A malicious participant can easily skew the election the way he wants by encrypting a number different from $-1, 1$. In Chapter 3 we will solve this problem using a Σ-protocol, while in Chapter 4 we will use verifiable secret sharing, in addition to a Σ-protocol, to make a voting system that preserves privacy in the information-theoretically sence.

2.6 Commitment schemes

The notion of commitments is at the heart of almost any construction of modern cryptographic protocols [Dam99]. In Chapter 3 we use commitments to improve the efficiency of the needed Σ-protocol for proving correctness of the vote.

Making a commitment simply means that a player in a protocol is able to choose a value from some set and commit to his choice such that he can no longer change his mind. The player does not have to reveal his choice, although he may choose to do so at a later time. The following informal physical analogy from [Dam99] illustrates the properties that are essential in a commitment scheme.

Analogy. Player P wants to commit to some value s. To do so he writes down s on a piece of paper. He then puts the paper in a box which he locks and gives to player V. When V receives the box, he cannot tell what is inside before P decides to give him the key. This is called the *hiding property* of the commitment. In the meantime, having given away the box, P cannot anymore change what is inside. Hence, when the box is opened, we know that what is revealed really was the choice that P committed to originally. This is called the *binding property* of the commitment. If P wants to, he can later *open* the commitment by giving V the key to the box.

Definition. ([DGS03]) A *commitment scheme* Γ is a triple $\Gamma = (Gen, com, ver)$, where com and ver is a family of commitment and verification functions respectivly, and Gen is an efficient key generating algorithm which generates a key K. With this key K there is an associated message space $\mathcal{M}_K$, a randomizer space $\mathcal{R}_K$, a commitment space $\mathcal{C}_K$, an opening space $\mathcal{B}_K \supseteq \mathcal{R}_K$, a commitment function $com_K(\cdot,\cdot) : \mathcal{M}_K \times \mathcal{R}_K \to \mathcal{C}_K$, and a verification function $ver_K(\cdot,\cdot,\cdot) : \mathcal{M}_K \times \mathcal{B}_K \times \mathcal{C}_K \to \{0,1\}$.

Given the key, a commitment to $m \in \mathcal{M}_K$ can be formed by choosing a random $r \in \mathcal{R}_K$ and computing $c = com(m;r)$. The triple (m, r, c) satisfies $ver_K(m;r;c) = 1$. While, to open a commitment $c \in \mathcal{C}_K$ an element $m \in \mathcal{M}_K$, $r \in \mathcal{B}_K$ is revealed such that $ver(m;r;c) = 1$.

The binding and hiding property of commitment schemes comes in multiple flavors. A commitment scheme is said to be *unconditionally binding* if even a prover with unlimited computer power cannot change his mind after forming the commitment. On the other hand, a commitment scheme is said to be *computational binding* if the probability of finding a commitment in $\mathcal{C}_K$ and two correct openings of it with different messages m_1 and m_2 is negligible in the security parameter[8].

Similarly is a commitment scheme said to be *unconditionally hiding* if given m the distribution of $com_K(m;r)$ is uniform in $\mathcal{C}_K$. A commitment scheme is said to be *statistically hiding* if the distribution is close to uniform $\mathcal{C}_K$. That is, with some negligible probability greater then $1/|\mathcal{C}_K|$ the verifier is able to guess m given $com_K(m;r)$. And finally, a commitment scheme is said to be *computationally hiding* if the probability of finding a correct opening for a given commitment is negligible in the security parameter[9].

[8]More formally, take any probabilistic polynomial time algorithm P^* which takes as input a public key generated by Gen on input 1^l. Let $\epsilon(l)$ be the probability (over random choices of Gen and P^*) the algorithm outputs a commitment and two valid openings revealing distinct values. Then the commitment scheme is computational binding if $\epsilon(l)$ is *negligible* in l.[Dam99]

[9]See [Dam99] for a formal definition of this in the bit-commitment scheme scenario.

Note that a commitment scheme can not be unconditionally biding and unconditionally hiding at the same time. We refer the interested reader to [Dam99] for an easy explanation of the reason for this limitation.

3

An electronic voting system based on homomorphic encryption

3.1 Introduction

We will now present an electronic voting system based on homomorphic encryptions. The voting system we present is from “The theory and implementation of an electronic voting system,” by Ivan Damgård, Jens Groth and Gorm Salomonsen [DGS03].

Resume of the electronic voting system based on homomorphic encryption.

Set-up. Let M be a strict upper bound of the number of voters and let L be the number of candidates. A vote on candidate number $j \in \{0, \dots, L-1\}$ is represented as the number M^j. We assume we are in possession of a homomorphic encryption scheme $\Lambda = (Gen, E, D)$, where a key pk already has been generated.

Voting phase. Each voter i encrypts his vote $y_i = E_{pk}(M^{j_i}; r_i)$ and uses a Σ-protocol to prove that y_i indeed is a correctly formed vote.

Tallying. The voting authorities computes

$$y = \prod_{i \in A} y_i = E_{pk}(v_0 M^0 + \cdots + v_{L-1} M^{L-1}; r),$$

where A is the collection of accepted votes, v_j is the number of votes on candidate j and $r = \sum_{i \in A} r_i$. Finally y is decrypted to reveal the results of the election.

The voting system is much like the scheme in the example of Section 2.5. The voter uses a public-key cryptosystem to encrypt his vote which is represented as a number. When submitting his encrypted vote he must prove knowledge of the fact that his encryption contains a valid vote. Because all individual votes will remain encrypted and the proof is zero-knowledge, this does not violate privacy. On the other hand, since we use a homomorphic encryption scheme, the votes can be tallied efficiently. By computing the product of all the encrypted votes we get an encryption of the result, so finally all that is needed is to decrypt this.

Of course, we can not let a trusted center decrypt the results since this would violate universal verifiability. To solve this, the private key needed for decryption can be secretly shared among a set of n servers (held by n different authorities) using a (n, t)-threshold secret sharing scheme. This means that as long as at most $t-1$ servers are corrupt, or broken into by a hacker, no information about the private key leaks. On the other hand, if at least t servers behave correctly, the serves can jointly execute the decryption operation. This is known as *threshold decryption*[DGS03].

If we set $t = \lfloor (n-1)/2 \rfloor$, then we are guaranteed that if a majority of the servers are in operation and are not corrupted, the election results, and only that will be decrypted. In practice, one may imagine that some public institutions and political parties could be running these servers in order to create broad trust in the process [DGS03].

In the next section we present the building blocks need for both the voting system and the Σ-protocol, while in Section 3.3 we construct the needed Σ-protocol.

3.2 Building blocks

3.2.1 The commitment scheme

Following the results of [Lip01] we will use a commitment scheme to make the Σ-protocol for proving correctness of an encrypted vote more efficient. The use of an commitment scheme has two natural advantages. First, since the commitment scheme (in contrast to the encryption scheme) does not have to be unconditionally binding, they can be much easier to work with. Second, if we use an integer[1] commitment scheme we can potentially use special properties of $\mathbb{Z}$. In our case we will be using the property of unique factorization.

We will now present what properties the commitment scheme we need must satisfy. Note that the reason we need some of the properties below will not be clear until we use them in our Σ-protocol. First there is a key generation phase in which a public key is generated. In our case the election authorities will be the ones generating the key, but for now we will just assume that some public key K has been generated. With this key comes the associated spaces $\mathcal{M}_K, \mathcal{C}_K, \mathcal{R}_K$ and $\mathcal{B}_K \supset \mathcal{R}_K$, the commitment function $com_K(\cdot, \cdot)$ and the verification function $ver_K(\cdot, \cdot, \cdot)$. Note that we do allow for openings not corresponding to correctly formed commitments since the opening space and the randomizer space are not equal. We do demand that the scheme is computationally binding however.

In order for the commitment scheme to be useful in our voting scheme it must in addition meet the following requirements.

Abelian groups. The spaces associated with the commitment scheme must be Abelian groups, and the message space must be the set of integers. That means we have groups $\mathcal{M}_K = \mathbb{Z}$, $(\mathcal{R}_K, +) \subset (\mathcal{B}_K, +)$ and $(\mathcal{C}_K, \cdot)$. We assume that both the group and the elements in the groups we work with can be represented in a suitable manner, that binary operations and inversions can be computed efficiently, and that we can readily recognize whether an element belongs to a particular group.

Homomorphic property. The commitment scheme we look at must be homomorphic, meaning that for all $m_1, m_2 \in \mathbb{Z}$ and all $r_1, r_2 \in \mathcal{B}_K$ the following is satisfied:

$$com_K(m_1; r_1) \cdot com_K(m_2; r_2) = com_K(m_1 + m_2; r_1 + r_2).$$

Root opening. Vi demand that our commitment scheme fulfills the following property: For any $c \in \mathcal{C}_K$, if we can find $e \in \mathbb{Z}\backslash\{0\}$ and $m \in \mathbb{Z}$, $z \in \mathcal{B}_K$ such that $com_K(m; z) = c^e$ then we can compute an opening of c.

An example of such a commitment scheme is the following variant of the Damgård-Fujisaki commitment scheme from [DF01]. Note that any commitment scheme that fulfills the above criteria can be used in our protocol.

[1] A commitment scheme is called an *integer* commitment scheme if the scheme allows for committing to arbitrary size integers.

Damgård-Fujisaki commitment scheme

Set-up. The receiver (election authorities) generates n as the product of two safe primes[2]. Let $ord(n) \leq 2^B$ and let k be our security parameter. The receiver then chooses a square h, and a random $\alpha \in [0, \dots, 2^{B+k}]$ and let $g = h^\alpha$. The key is $K = (n, g, h)$.

Commit. A commitment to an integer m is formed by choosing $r \in [0, \dots, 2^{B+2k}]$ at random and letting the commitment be $com_{(n,g,h)}(m; r) = g^m h^r \bmod n$.

Open. To open a commitment c we produce b, m, r such that $1 = b^2 \bmod n$ and $c = bg^m h^r$. An honest prover can always use $b = 1$.[3]

Theorem 3.1. Under the RSA assumption the Damgård-Fujisaki commitment scheme is statistically hiding and computational binding.

Proof. As for hiding, note that since r is chosen with bit-length at least twice that of order h, $com_K(m; r)$ is statistically close to uniform in $\langle h \rangle$ for any value m.

As for binding, suppose some prover P can produce a commitment c and two valid openings $(m, r, b), (m', r', b')$ with $m \neq m'$. We then have $g^m h^r b = c = g^{m'} h^{r'} b'$. Using $g = h^\alpha$ and squaring both sides we get

$$h^{2\alpha(m-m')+r-r'} = 1$$

Let $M = 2\alpha(m - m') + r - r'$, we will argue that $M \neq 0$. Since α is chosen to be much larger then the order of h it follows that P does not have full information on α. Even a prover with infinite computing power can only computer $\alpha \bmod ord(h)$. Let $\alpha = q \cdot ord(h) + res$. It follows that P's choice of m, m', r, r', is independent of q. Clearly, in order for M to be zero, q must be a particular value which only occurs with negligible probability since we choose α at random in a very large interval. Finally, if $M \neq 0$, we have a multiple of the order of h and this is enough to factor n, and thus solve the RSA problem [RK03]. □

3.2.2 The encryption scheme

We will now take a closer look on the encryption scheme we will be using in the voting system. Unlike the commitment scheme, the encryption scheme was designed particularly for this voting system. We will therefore follow the construction from [DGS03], and present the useful properties of the scheme afterwards.

Let R be a commuatativ ring, fix some $g \in R$ and let $G = \langle g \rangle$. We assume that one can compute addition and multiplication efficiently in R and that a number T can be computed easily, so that $T \geq ord(g)^2$. This just requires some upper bound on $ord(g)$ to be known. We will assume that the generalized Decision Diffie-Hellman assumption holds with respect to R and g. Remember that this means that given R, g, triples of the form g^a, g^b, g^{ab} are computationally indistinguishable from triples of the form g^a, g^b, g^c where a, b, c are random in $[0 \dots T]$. Note that the choice of T ensures that the distribution of elements such as g^a is statistically close to uniform in $\langle g \rangle$ as long as $ord(g)$ is large.

We can now define an ElGamal style cryptosystem where the private key is chosen at random in $[0 \dots T]$ and the public key is (R, g, h) with $h = g^x$. The message space is $\langle g \rangle$, and to encrypt a message m, choose $r \in [0 \dots T]$ at random and output $E(m; r) = (g^r; mh^r)$. To decrypt a ciphertext (u, v) we compute $D(u, v) = v(u^x)^{-1}$. This system is semantically secure[4] under the

[2]A prime p is called *safe* if $(p-1)/2$ also is a prime

[3]The reason we give the prover this extra freedom is that this makes it possible to construct a Σ-protocol for proving knowledge of commitment opening. We refer the interested reader to [DF01] for details.

[4]A cryptosystem is said to be semantically secure if a polynomial time adversary cannot with probability greater than 1/2 distinguish between the encryptions of two given plaintext, or the encryptions of a given plaintext and a random string.

generalized DDH assumption [DGS03].

This cryptosystem is not homomorphic however. To solve this problem we redefine the system by fixing an element $w \in \langle g \rangle$ and letting the message space be $\mathbb{Z}_{ord(w)}$. We can now define $E(m; r) = (g^r; w^m h^r)$. This does not effect the semantic security, but it does make the encryption homomorphic, since $E(m_1; r_1)E(m_2; r_2) = E(m_1 + m_2 \bmod ord(w); r_1 + r_2)$. Unfortunately, now we have a problem with decryption. Computing $D(u, v) = v(u^x)^{-1}$ yields w^m for a correctly formed encryption. So to decrypt a ciphertext we need to be able to compute discrete logarithms with base w.

The trick is that in some rings we can find certain elements for which computing discrete logarithms is in fact easy. Let $w = \alpha + \beta$, then

$$w^i = (\alpha + \beta)^i = \sum_{j=0}^{i} \binom{i}{j} \alpha^j \beta^{i-j}$$

using standard binomial expansion. Usually i is exponentially large, so this formula does not help to compute i. But if α is nilpotent, that is if $\alpha^j = 0$ for some (preferably small) j, then most of the terms in the expansion disappear, and it may therefore be feasible to compute i.

We will now give a concrete example of such an encryption scheme.

Encryption scheme

Set-up. Let $R = \mathbb{Z}_{n^{s+1}}$, where $n = pq$ is an RSA modulus, $gcd((p-1), (q-1)) = 2$ and $s << p, q$. Let $g \in R$ have maximal order, that is $ord(g) = n^s(p-1)(q-1)$. Set $w = n+1 \in R$, $T = ord(g)^2$ and $h = g^x$ for some random $x \in [0 \ldots T]$. The public key is R, g, h, w, while the private key is x.

Encryption. To encrypt a message $m \in \mathbb{Z}_{ord(w)}$ choose $r \in [0 \ldots T]$ at random and output $E(m, r) = (g^r, w^m h^r)$.

Decryption. To decrypt a ciphertext (u, v) compute $D(u, v) = \log_w(v(u^x)^{-1})$.

Clearly n is nilpotent in R since $n^s = 0$, so discrete logarithms base $w = n + 1$ are therefore feasible to compute by the discussion above. A concrete algorithm for this can be found in Appendix A. By Lemma 3.1 $ord(w) = n^s$, and the message space for the encryption scheme is therefore $\mathbb{Z}_{n^s}$.

Lemma 3.1. ([DJ01]) Let $n = pq$, where p, q are odd primes. For any $s < p, q$, the element $n+1$ has order n^s in $\mathbb{Z}^*_{n^{s+1}}$.

Proof. Consider the integer $(1 + n)^i = \sum_{j=0}^{i} \binom{i}{j} n^j$. This number is 1 modulo n^{s+1} for some i if and only if $\sum_{j=1}^{i} \binom{i}{j} n^{j-1}$ is 0 modulo n^s. Clearly, this is the case if $i = n^s$, so it follows that the order of $1 + n$ is a divisor of n^s, which means that the order is a number of the form $p^\alpha q^\beta$, where $\alpha, \beta \leq s$.

Now set $a = p^\alpha q^\beta$, and consider a term $\binom{a}{j} n^{j-1}$ in the sum $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}$. We claim that each such term is divisible by a. This follows from the fact that if $j > s$, then a divides n^{j-1}, while if $j \leq s$, then $j!$ can not have p or q as prime factors, and so a must divide $\binom{a}{j}$.

Now assume for contradiction that $a = p^\alpha q^\beta < n^s$. Without loss of generality, we can assume this means $\alpha < s$. We know that n^s divides $\sum_{j=1}^{a} \binom{a}{j} n^{j-1}$. Dividing both numbers by a we see that p must divide the number $\sum_{j=1}^{i} \binom{a}{j} n^{j-1}/a$. However, the first term in this sum after division by a is 1, and all the rest are divisible by p, so the number is in fact 1 modulo p, and we have a contradiction. $\square$

Theorem 3.2. Under the generalized Decision Diffie Hellman assumption the encryption scheme above is semantically secure.

Proof. Let m be a plaintext and m' a random string. Then

$$E(m; r) = (g^r; w^m g^{xr}),$$

while, using that $w \in \langle g \rangle$,

$$E(m'; r) = (g^r; w^{m'} g^{xr}) = (g^r; w^m g^c),$$

for some $c \leq ord(g)$. It follows that if an adversary could distinguish between encryptions corresponding to m and m' respectively, he would also be able to solve the generalized Decision Diffie Hellman problem on input (g^x, g^r, g^c). $\square$

Note that a total break of the encryption relies on the discrete logarithm assumption. If an adversary is able to compute $\log_g h$, he will be able to decrypt any given ciphertext.

The threshold decryption only requires that we can compute securely $u^x \bmod n^{s+1}$ given u and a secret sharing of x. Damgård et. al. [DGS03] proposes to use the protocol found in [DJ01]. Unfortunately this protocol is not directly applicable. We instead propose the following similar protocol, where (u, v) is the ciphertext, x is the private key, l is the number of authorities and t is our threshold.

Threshold decryption

Set-up. The following is appended to the set-up of the encryption scheme above. Using Shamir's secret sharing scheme from Section 2.4, we chose at random $f(X) = \sum_{i=0}^{t-1} \rho_i X^i \bmod n^s(p-1)(q-1)$ by setting $\rho_0 = x$ and the remaining ρ_i to random elements from $[0 \dots T]$. The secret share of the i'th authority will be $s_i = f(i)$ for $0 \leq i \leq l$. For verification of the action of the authorities we need the following fixed public values; for each authority i a verification key $\eta_i = g^{s_i}$.

Share decryption. Upon receiving (u, v) each authority i computes $u_i = u^{s_i}$. Then authority i publishes u_i along with a proof that $\log_u(u_i)$=$\log_g(\eta_i)$ using a Fiat Shamir heuristic non-interactive Σ-protocol. This will convince us that authority i indeed raised to his secret exponent[5].

Share combination. If we have the required t (or more) number of shares with a correct proof, we can combine them into the result by taking a subset S of t shares an combine them to

$$u' = \prod_{i \in S} u_i^{\lambda_{S,i}}, \quad \text{where } \lambda_{S,i} = \prod_{j \in S \setminus \{i\}} \frac{-j}{i-j}.$$

Theorem 3.3. The threshold decryption above outputs the desired u^x if t or more authorities follow the protocol. Assuming the random oracle model and the discrete logarithm assumption no set of $t-1$ authorities or less can compute u^x.

Proof. First we show that u^x is computed correctly. Let S be a set of t or more correct shares, then

$$u' = \prod_{i \in S} u_i^{\lambda_{S,i}} = \prod_{i \in S} (u^{s_i})^{\lambda_{S,i}} = \prod_{i \in S} u^{s_i \lambda_{S,i}} = u^{\sum_{i \in S} s_i \lambda_{S,i}} = u^{f(0)} = u^x,$$

by the discussion in Section 2.4.

[5] A Σ-protocol for proving equality of discrete logarithms can easily be derived from Schnorr's protocol. We give a discription in Appendix B

On the other hand in the random oracle model the non-interactive Σ-protocol is zero-knowledge thereby releasing no information about individual shares to corrupt authorities. Furthermore, no adversary can compute $s_i = \log_g \eta_i$ under the discrete logarithm assumption. It follows from the discussion in Section 2.4 that a set of $t-1$ authorities have no information on x. This implies that no set of $t-1$ authorities or less can compute u^x. $\square$

We note that in order the prove that the threshold decryption gives *no* information about the content of the encryption we must (if possible) show that, from the view of the corrupted authorities, the key generation and the decryption protocol can be efficiently simulated with a statistically indistinguishable distribution. Unfortunately this is outside the scope of this thesis.

The encryption scheme satisfies the following properties which we will note for later.

Homomorphic property. The encryption scheme is homomorphic, that is $E(m_1; r_1)E(m_2; r_2) = E(m_1 + m_2 \bmod w; r_1 + r_2)$ for all $m_1, m_2 \in \mathcal{M}_{pk}$ and all $r_1, r_2 \in \mathcal{R}_{pk}$

Root opening. The encryption scheme satisfies a weaker root opening property then the commitment scheme. Given a valid ciphertext (u, v) we may extract the plaintext of (u, v) from an opening of (u^e, v^e), where $0 < e < p, q$. The reason for this is that the plaintext corresponding to (u^e, v^e) must be $em \bmod n^s$, and so we can find m because e is always invertible modulo n^s.

In the upcoming construction of the Σ-protocol, any public key cryptosystem with threshold decryption that satisfies these properties will work, even if the root opening property is only satisfied for $e \in \{0, \ldots, 2^t - 1\}$, where t is some security paramete. Therefore, we describe the protocols in general terms in what follows. We shall write *pk* for the public key of the cryptosystem, and let $\mathcal{C}_{pk}$ be the corresponding ciphertextspace, consisting of only valid ciphertexts.

3.3 Constructing the Σ-protocol

Looking closer at the schemes proposed in literature, the really heavy part of generating a vote and tallying a vote, both in term of communication complexity and computational complexity, is producing and verifying the zero-knowledge proof associated with it [DGS03]. Therefore, constructing an efficient Σ-protocol is a vital part in the construction of efficient electronic voting systems based on homomorphic encryptions.

As mentioned in Section 3.2.1, we will follow [Lip01] and use commitments to prove the correctness of an encrypted vote and then prove that the encryption and commitment hold the same element. To prove the latter we can use the following Σ-protocol from [DGS03].

Proof of commitment and encryption holding same element modulo n

Common input: A commitment $c \in \mathcal{C}_K$ and an encryption $y \in \mathcal{C}_{pk}$.
Private input for the prover: $m \in \mathbb{Z}_n$, $r_c \in \mathcal{R}_K$ and $r_y \in \mathcal{R}_{pk}$ so that $c = com_K(m; r_c)$ and $y = E_{pk}(m; r_y)$.

Initial message. Pick $d \in \mathbb{Z}$ as a shadow[6]of em, $r'_c \in \mathcal{R}_K$ as a random shadow of er_c and $r'_y \in \mathcal{R}_{pk}$ as a random shadow of er_y. Let $a_c = com_K(d; r'_c)$ and $a_y = E_{pk}(d \bmod n; r'_y)$. The initial message is (a_c, a_y).

Challenge. Select at random $e \in \{0, \ldots, 2^t - 1\}$.

Answer. Set $D = em + d$, $z_c = r'_c + er_c$, $z_y = r'_y + er_y$. The answer to the challenge is (D, z_c, z_y).

Verification. The verifier checks that $(D, z_c, z_y) \in \mathbb{Z} \times \mathcal{R}_K \times \mathcal{R}_{pk}$, $com_K(D; z_c) = a_c c^e$ and $E_{pk}(D \bmod n; z_y) = a_y y^e$.

[6]Remember that by the definition of shadowing we want to chose d such that revealing $d + em$ does not give

Assume $c = com_K(m;\cdot)$, and $y = E_{pk}(m';\cdot)$ with $m \neq m'$. Informally the protocol works by the observation that if the commitment and encryption holds different elements, then either $a_c c^e \neq com_K(d+em;\cdot)$ or $a_y y^e \neq E_{pk}(d+em';\cdot)$. A proof that this is a Σ-protocol can be found in [Lip01].

Now that we can prove that the commitment and encryption hold the same element modulo n, the next question is how to prove that the commitment contains a vote on the correct form. Remember that to vote on candidate $j \in \{0,\ldots,L-1\}$, the voter encrypts the number M^j. The idea is to let M be the square of a prime, $M = p^2$. We can now prove that a commitment c_{M^j} to M^j is on the correct form by the following. First show that c_{M^j} contains the square of a commitment c_{p^j} to p^j, and then prove that the contents of c_{p^j} multiplied with the content of another commitment $c_{p^{L-j-1}}$ equals p^{L-1}. To do this with need a protocol for proving that given commitments c_a, c_b, c_c the contents of c_a multiplied with the contents of c_b equals the contents of c_c. The following protocol from [DGS03] accomplishes this.

Proof of multiplicative relationship

Common input: $c_a, c_b, c_c \in \mathcal{C}_K$.
Private input for the prover: $a, b \in \mathbb{Z}$, $r_a, r_b, r_c \in \mathcal{R}_K$ such that $c_a = com_K(a; r_a)$, $c_b = com(b; r_b)$, $c_c = com_K(ab; r_c)$.

Parallel proof. Make in parallel with the rest of the protocol a proof of knowledge of commitment opening of c_b or c_c using a Σ-protocol[7].

Initial message. Select $d \in \mathbb{Z}$ such that it shadows ea. Choose $r_d, r_{db} \in \mathcal{R}_K$ as random shadows of er_a and $-(ea+d)r_b + er_c$, and send $c_d = com_K(d; r_d)$ and $c_{db} = com_K(db; r_{db})$ to the verifier.

Challenge. Select at random $e \in \{0,\ldots,2^t-1\}$.

Answer. Respond with $f = ea + d$, $z_1 = er_a + r_d$, $z_2 = fr_b - er_c - r_{db}$.

Verification. Accept if and only if $f \in \mathbb{Z}$, $z_1, z_2 \in \mathcal{R}_K$, $com_K(f; z_1) = c_d c_a^e$ and $c_{db} c_c^e com_K(0; z_2) = c_b^f$ and the parallel proof of knowledge is acceptable.

Informally the protocol works by the observation that if $c \neq ab$ then $c_b^f = com_K(b(ea+d);\cdot) = com_K(ab;\cdot)^e com_K(bd;\cdot) com_K(0;\cdot) \neq com_K(c;\cdot)^e com_K(db;\cdot) com_K(0;\cdot) = c_c^e c_{db} com_K(0;\cdot)$. A proof that this is a Σ-protocol can be found in [DJ01].

To prove that an encrypted vote is on the correct form we can now do the following:

1. Encrypt $y_{M^j} = E_{pk}(M^j;\cdot)$ and form commitments $c_{M^j}, c_{p^j}, c_{p^{L-1}}, c_{p^{L-j-1}}$ to M^j, p^j, p^{L-1}, p^{L-j-1} respectively.
2. Use the proof of multiplicative relationship to prove that the the square of the content of c_{p^j} equals the content of c_{M^j}.
3. Use the proof of multiplicative relationship to prove that the content of c_{p^j} times the content of $c_{p^{L-j-1}}$ equals the content of $c_{p^{L-1}}$. This shows that the content of c_{p^j} is a power of p less or equal to $L-1$.
4. Finally prove that y_{M^j} and c_{M^j} hold the same element modulo n.

We can make a few improvements to increase the efficiency of the proof. First note that we can skip the parallel proof of some opening in the proof of multiplicative relationship. The reason

away any information about m. Here we know that $e \in \{0,\ldots,2^t-1\}$ while $m \in \mathbb{Z}_n$. Thus by selecting d at random from $\{-2^{|n|+2t},\ldots,2^{|n|+2t}\}$ we ensure that the secrecy of m is preserved.

[7]A Σ-protocol for proving knowledge of commitment opening can be found in [DF01].

for this is that we in step 3 do a proof of multiplicative relationship where we already know the opening for $c_{p^{L-1}} = com_K(p^{L-1}; 0)$. The price for this improvement is that we require the root opening property to be stronger then usual. We require that knowing an opening of c^f with $f \in \mathbb{Z}\backslash\{0\}$ makes it possible to find an opening of c. While usually, one only requires that knowing an opening of c^e for a commitment c with $e \in \{1, \ldots, 2^t - 1\}$ makes it possible to open c [DGS03].

As a second improvement note that the commitment c_{p^j} is used in both the multiplication proof in step 2 and in the multiplication proof in step 3. Since the proof system is special honest verifier zero-knowledge, we can choose the same challenge e in both these proofs. That means we can recycle the values d, c_d, f and z_1 to improve efficiency.

As a last improvement note that we do not really need a commitment to M^j. Instead we can prove directly that the encryption of the vote holds the same element as the square of the contents of c_{p^j}. The reason we can do this improvement is that, on the commitment side in step 2, we use c_{M^j} to hold the square of the contents of c_{p^j}. But instead of using c_{M^j}, we can use $c_{p^j}^f = c_{p^j}^{ep^j+d}$ directly since, by the homomorphic property, this is a commitment to $ep^{2j} + dp^j$ and therefore contains the interesting p^{2j} itself. Combining these three improvements we get the following protocol from [DGS03].

Proof of knowledge for a ciphertext containing a valid vote

Common input: Prime p such that $M = p^2$ and an encryption $y \in \mathcal{C}_{pk}$.
Private input for the prover: $0 \leq j \leq L$ and $r_y \in \mathcal{R}_{pk}$ such that $y = E_{pk}(M^j; r_y)$.

Initial message. Choose first r_a, r_b at random from $\mathcal{R}_K$ and form commitments $c_a = com_K(p^j; r_a)$ and $c_b = com(p^{L-j-1}; r_b)$.
Choose $d \in \mathbb{Z}$ such that i shadows p^j. Choose γ such that it shadows $eM^j + dp^j$. Choose $r_d, r_{db}, r_\gamma \in \mathcal{R}_K$ and $r'_\gamma \in \mathcal{R}_{pk}$ as random shadows of er_a, $(ep^j + d)r_b$, $(ep^j + d)r_a$ and er_y respectively.
Send $c_d = com_K(d; r_d)$, $c_{db} = com_K(dp^{L-j-1}; r_{db})$, $c_\gamma = com_K(\gamma; r_\gamma)$ [8]and $y_\gamma = E_{pk}(dp^j + \gamma \bmod n; r'_\gamma)$ to the verifier.

Challenge. Select at random $e \in \{0, \ldots, 2^t - 1\}$.

Answer. Send $f = ep^j + d$, $z_1 = er_a + r_d$, $z_2 = fr_b - r_{db}$, $z_3 = fr_a + r_\gamma$, $z_4 = er_y + r'_\gamma$ and $D = eM^j + dp^j + \gamma$ to the verifier.

Verification. Check that $c_d, c_{db}, c_\gamma \in \mathcal{C}_K$, $f, D \in \mathbb{Z}$, $z_1, z_2, z_3 \in \mathcal{R}_K$ and $z_4 \in \mathcal{R}_{pk}$.
Verify that $com_K(f; z_1) = c_d c_a^e$, $c_{db} com_K(p^{L-1}; 0)^e com_K(0; z_2) = c_b^f$, $com_K(D; z_3) = c_a^f c_\gamma$ and $E_{pk}(D; z_4) = y^e y_\gamma$.

Theorem 3.4. The proof system above is a Σ-protocol for proving that y is a ciphertext holding a vote of the correct form. It is statistical special honest verifier zero-knowledge if the commitment scheme is statistically hiding and the shadows are statistically hiding.

Proof. Theorem 3.4 follows as a corollary to Theorem 3.5 proven later. $\square$

One of the advantages of the approach to building the Σ-protocol above is that we can extend it to the situation where each voter is allowed to cast several votes on different candidates. Let N be the number of candidates a voter may vote for. Notice that it is sufficient for a voter to give an encryption of the sum of the votes and then prove that the sum is correct. That is, if we write the candidates in increasing order $0 \leq j_1 < \cdots < j_N < L$, we can encrypt $y = E_{pk}(M^{j_1} + \cdots + M^{j_N}; \cdot)$,

[8]We have here corrected a typographical error in [DGS03].

and then use a Σ-protocol that proves that y is on the correct form. To prove this we can do the following:

1. Encrypt $y = E_{pk}(M^{j_1} + \cdots + M^{j_N}; \cdot)$ and form commitments $c_1, \ldots, c_n$ to $p^{j_1}, \ldots p^{j_N}$, $c'_1, \ldots c'_N$ to $p^{j_2-j_1-1}, \ldots p^{L-j_N-1}$ and $c''_1, \ldots, c''_N$ to $M^{j_1}, \ldots, M^{j_N}$.

2. Use the proof of multiplicative relationship to prove that the square of the content of c_i equals the content of c''_i for $i = 1 \ldots N$.

3. Use the proof of multiplicative relationship to prove that the content of $c_i(c'_i)^p$ equals the content of c_{i+1} for $i = 1 \ldots N$, where $c_{N+1} = com_K(p^L; 0)$. This proves that, up to a difference in sign, all commitments are different, all contain a power of p and all exponents lie in the interval $[0, \ldots, L-1]$.

4. Compute the commitment $c''_1 \cdots c''_N$. This is a commitment to a vote which we have shown is on the form $M^{j_1} + \cdots + M^{j_N}$, where $0 \leq j_1 < \ldots j_N < L$. Finally, use the proof of multiplicative relationship to prove that the content of y equals the content of $c''_1 \cdots c''_N$.

We can make similar improvements as we did for the single vote scenario. First note that we can skip the parallel proof of some opening in the proofs of multiplicative relationship. The reason we can do this is that we already know the opening for $com_K(p^L; 0)$, and in the multiplication proof to $c_N(c'_N)^p = c_{N+1}$ we show that we know an opening of c_N and so on.

For the second improvement we note that the commitments $c_1, \ldots, c_N$ are being used in two proofs of multiplicative relationship. This means that we can make the protocol more efficient by using the same challenge e in all the proofs, since this allows us to recycle the d, c_d, f, z_1 parts of the multiplication proofs.

As a last improvement we note that since we use the same challenge e in all the proofs we can avoid supplying the commitments $c''_1, \ldots c''_N$ in a similar manner as the single candidate scenario. Combining all these ideas we get the following protocol from [DGS03].

Proof of knowledge for a ciphertext containing a valid vote on multiple candidates

Common input: Prime p such that $M = p^2$ and $y \in \mathcal{C}_{pk}$.
Private input for the prover: $0 \leq j_1 < \cdots < j_N < L$ and $r_y \in \mathcal{R}_{pk}$ such that $y = E_{pk}(M^{j_1} + \cdots + M^{j_N}; r_y)$.

Initial message. Choose at random $r_1, \ldots, r_N, r'_1, \ldots, r'_N$ from $\mathcal{R}_K$, and form commitments $c_1 = com_K(p^{j_1}; r_1), \ldots, c_N = com_K(p^{j_N}; r_N)$, $c'_1 = com_K(p^{j_2-j_1-1}; r'_1), \ldots, c'_N = com_K(p^{L-j_N-1}; r'_N)$.
Choose $d_1, \ldots, d_N \in \mathbb{Z}$ such that they shadow $ep^{j_1}, \ldots, ep^{j_N}$, and $\gamma \in \mathbb{Z}$ such that it shadows $ep^{2j_1} + d_1 p^{j_1} + \cdots + ep^{2j_N} + d_N p^{j_N}$.
Choose $r_{d_1}, \ldots, r_{d_N} \in \mathcal{R}_K$ as random shadows of $er_1 \ldots, er_N$. Choose $r_{1b}, \ldots r_{d_N b} \in \mathcal{R}_K$ as random shadows of $-p(ep^{j_1} + d_1)r'_1 + er_2, \ldots, -p(ep^{j_N} + d_N)r'_N + er_{N+1}$, where $r_{N+1} = 0$. Choose $r_\gamma \in \mathcal{R}_K$ as a random shadow of $(ep^{j_1} + d_1)r_1 + \cdots + (ep^{j_N} + d_N)r_N$, and $r'_\gamma \in \mathcal{R}_{pk}$ as a random shadow of er_y.
Send $c_{d_1} = com_K(d_1; r_{d_1}), \ldots, c_{d_N} = com_K(d_N; r_{d_N})$, $c_{d_1 b} = com_K(d_1 p^{j_2 - j_1}; r_{d_1 b}), \ldots, c_{d_N b} = com_K(d_N p^{L-j_N}; r_{d_n b})$, $c_\gamma = com_K(\gamma; r_\gamma)$ and $y_\gamma = E_{pk}(d_1 p^{j_1} + \cdots + d_N p^{j_N} + \gamma \bmod n; r'_\gamma)$ to the verifier.

Challenge. Select at random $e \in \{0, \ldots, 2^t - 1\}$.

Answer. Send $f_1 = ep^{j_1} + d_1, \ldots, f_N = ep^{j_N} + d_N$, $z_{1,1} = er_1 + r_{d_1}, \ldots, z_{1,N} = er_N + r_{d_N}$, $z_{2,1} = pf_1 r'_1 - er_2 - rd_{1b}, \ldots, z_{2,N} = pf_N r'_N - er_{N+1} - rd_{d_N b}$, $z_3 = f_1 r_1 + \cdots + f_N r_N + r_\gamma$, $z_4 = er_y + r'_\gamma$, $D = e(M^{j_1} + \cdots + M^{j_N}) + d_1 p^{j_1} + \cdots + d_N p^{j_N} + \gamma$ to the verifier.

Verification. Check that $c_{d_1}, \ldots, c_{d_N}, c_{d_1 b}, \ldots, c_{d_N b}, c_\gamma \in \mathcal{C}_K$, $y_\gamma \in \mathcal{C}_{pk}$, $f_1, \ldots, f_N, D \in \mathbb{Z}$, $z_{1,1}, \ldots, z_{1,N}, z_{2,1}, \ldots, z_{2,N}, z_3 \in \mathcal{R}_K$ and $z_4 \in \mathcal{R}_{pk}$. Verify that $com_K(f_1; z_{1,1}) = c_{d_1} c_1^e, \ldots, com_K(f_N; z_{1,N}) = c_{d_N} c_N^e$, $c_2^e c_{d_1 b} com_K(0; z_{2,1}) = (c'_1)^{pf_1}, \ldots, c_{N+1}^e c_{d_N b} com_K(0; z_{2,N}) = (c'_N)^{pf_N}$, $com_K(D; z_3) = c_1^{f_1} \cdots c_N^{f_N} c_\gamma$, where $c_{N+1} = com_K(p^L; 0)$. Finally check that $E_{pk}(D \bmod n; z_4) = y^e y_\gamma$.

Note that we can make the protocol above non-interactive by using the Fiat-Shamir heuristics discussed in Section 2.3.3. Given a suitible strong cryptographic hash-function $\mathcal{H}$ thought of as a random oracle, the prover uses $e = \mathcal{H}(c_{d_1}, \ldots, c_{d_N}, c_{d_1 b}, \ldots, c_{d_N b}, c_\gamma, y_\gamma)$ as the challange to compute to the answer z. The prover then sends (a, z) to the verifer who uses the above verification procedure to check if $(a, \mathcal{H}(c_{d_1}, \ldots, c_{d_N}, c_{d_1 b}, \ldots, c_{d_N b}, c_\gamma, y_\gamma), z)$ is an accepting conversation. To prevent vote duplication, a bit string specific to each voter can also be included in the hash function. In case the proof is done interactivly this bit string could be incorporated in the challange.

Theorem 3.5. The proof system above is a Σ-protocol proving that y encrypts a correct vote on multiple candidates. If the commitments are statistically hiding and the shadows and random shadows are statistically hiding, then the proof system is statistical honest verifier zero-knowledge.

Proof. We want to show that the protocol above satisfies the definition of a Σ-protocol with respect to completeness, special soundness and special honest verifier zero-knowledge.

Completeness. We want to show that if the prover is honest a honest verifier will always accept. So, assume the prover is honest, then

$$com_K(f_i; z_{1,i}) = com(ep^{j_i} + d_i; er_i + r_{di}) = com_K(p^{j_i}; r_i)^e com_K(d_i; r_{d_i}) = c_i^e c_{d_i},$$

for all i,

$$\begin{aligned} c_{i+1}^e c_{d_i b} com_K(0; z_{2,i}) &= com_K(p^{j_{i+1}}; r_{i+1})^e com_K(d_i p^{j_{i+1} - j_i}; r_{d_i b}) com_K(0; pf_i r'_i - er_{i+1} - r_{d_i b}) \\ &= com_K(ep^{j_{i+1}} + d_i p^{j_{i+1} - j_i}; pf_i r'_i) com_K(p^{j_{i+1} - j_i - 1} p(ep^{j_i + 1} + d_i); pf_i r'_i) \\ &= com_K(p^{j_{i+1} - j_i - 1} pf; pf_i r'_i) = com_K(p^{j_{i+1} - j_i - 1}; r'_i)^{pf_i} = (c'_i)^{pf_i}, \end{aligned}$$

for all i,

$$\begin{aligned} com_K(D;z_3) &= com_K(e(M^{j_1}+\cdots+M^{j_N})+d_1p^{j_1}+\cdots+d_Np^{j_N}+\gamma; f_1r_1+\cdots+f_Nr_N+r_\gamma) \\ &= com_K(p^{j_1}(ep^{j_1}+d_1)+\cdots+p^{j_1}(ep^{j_1}+d_1)+\gamma; f_1r_1+\cdots+f_Nr_N+r_\gamma) \\ &= com_K(p^{j_1}f_1+\cdots+p^{j_N}f_N+\gamma; f_1r_1+\cdots+f_Nr_N+r_\gamma) = c_1^{f_1}\cdots c_N^{f_N}c_\gamma, \end{aligned}$$

and

$$\begin{aligned} E_{pk}(D \bmod n; z_4) &= E_{pk}(e(M^{j_1}+\cdots+M^{j_N})+d_1p^{j_1}+\cdots+d_Np^{j_N}+\gamma \bmod n; er_y+r_{\gamma'}) \\ &= E_{pk}(M^{j_1}+\cdots+M^{j_N};r_y)^e E_{pk}(d_1p^{j_1}+\cdots+d_Np^{j_N}+\gamma; r_{\gamma'}) = y^e y_\gamma. \end{aligned}$$

So completeness trivially holds with probability 1.

Special soundness. We want to show that from two acceptable answers to two different challenges we can find $0 \leq j_1 < \cdots < j_N < L$ and $r_y \in \mathcal{R}_{pk}$ such that $y = E_{pk}(M^{j_1}+\cdots+M^{j_N};r_y)$. So let $f_1,\ldots,f_N, z_{1,1},\ldots,z_{1,N}, z_{2_1},\ldots,z_{2,N}, z_3, D, z_4$ and $f'_1,\ldots,f'_N, z'_{1,1},\ldots,z'_{1,N}, z'_{2_1},\ldots,z'_{2,N}, z'_3, D', z'_4$ be answers to challenges e and e', both satisfying the criteria specified in the verification step. On the encryption side we then have

$$E_{pk}(D;z_4) = y^e y_\gamma \text{ and } E_{pk}(D';z'_4) = y^{e'} y_\gamma,$$

which gives

$$E_{pk}(D-D';z_4-z'_4) = y^{e-e'}.$$

By the root-opening property we can now find a plaintext for y. Let v be this plaintext.

On the commitment side we have

$$com_K(f_1;z_{1,1}) = c_{d_1}c_1^e \text{ and } com_K(f'_1;z'_{1,1}) = c_{d_1}c_1^{e'},$$

which gives

$$com_K(f_1-f'_1;z_{1,1}-z'_{1,1}) = c_1^{e-e'}.$$

By the root-opening assumption of the commitment scheme we can now find an opening of c_1. By the same argument we can also find an opening of $c_2,\ldots c_N$. Let $a_1,\ldots,a_N$ denote the content of these commitments.

From the second part of the multiplication proof we have

$$c_{N+1}^e c_{Nb} com_K(0;z_{2,N}) = (c'_N)^{pf_N} \text{ and } c_{N+1}^{e'} c_{Nb} com_K(0;z'_{2,N}) = (c'_N)^{pf'_N},$$

which gives

$$com_K(0;z_{2,N}-z'_{2,N}) com_K(p^L;0)^{e-e'} = (c'_N)^{f_N-f'_N}$$

We now know an opening of the commitment on the left hand side. We also know that $f'_N \neq f_N$ since $1 = com_K(0;0)$ and the left side can not be opened to zero by the binding property. Therefore, by the root-opening assumption, we can find an opening of c'_N. Moreover, the content of c'_N is non-zero since the left side opens to something non-zero. We can now, in a similar manner, go backwards and find an opening of $c'_{N-1},\ldots,c'_1$. Let $b_N,\ldots,b_1$ denote the contents of these commitments.

We now have openings for the commitments $c_1,\ldots,c_N, c'_1,\ldots,c'_N$ and y. Furthermore, by the binding property of the commitment scheme, these openings must be the only ones that the prover can produce. What is left to argue is that the opening of the encryption satisfies the requirements of the proof. In that case, we have extracted a witness for the vote being on the correct form.

We get from

$$com_K(f_N-f'_N;z_{1,N}-z'_{1,N}) = c_N^{e-e'},$$

that

$$f_N - f'_N = a_N(e-e') \Rightarrow \frac{f_N-f'_N}{e-e'} \in \mathbb{Z}.$$

From
$$(c'_N)^{f_N - f'_N} = com_K(0; z_{2,N} - z'_{2,N})com_K(p^L; 0)^{e-e'},$$
we see that
$$p(f_N - f'_N)b_N = (e - e')p^L.$$
This implies that
$$a_N b_N = p^{L-1},$$
which means that $|a_N| = p^{j_N}$ for some $0 \leq j_N < L$. In a similar fashion we deduce that $|a_1| = p^{j_1}, \ldots, |a_{N-1}| = p^{j_{N-1}}$ with $0 \leq j_1 < \cdots < j_{N-1} < j_N$.

We now proceed to the link between the commitments and the encryption. We have
$$com_K(D; z_3) = c^{f_1} \cdots c^{f_N} \text{ and } com_K(D'; z'_3) = c^{f'_1} \cdots c^{f'_N}$$
which implies that
$$com_K(D - D'; z_3 - z'_3) = c^{f_1 - f'_1} \ldots c^{f_N - f'_N}$$
Using the fact that for all i we have $a_i = \frac{f_i - f'_i}{e - e'}$, the above equation gives us
$$D - D' = (f_1 - f'_1)a_1 + \cdots + (f_N - f'_N)a_N = (e - e')(a_1^2 + \cdots + a_N^2) = (e - e')(p^{2j_1} + \cdots + p^{2j_N}).$$

On the encryption side the equation
$$E_{pk}(D - D'; z_4 - z'_4) = y^{e-e'},$$
shows that v, the content of y, satisfies
$$D - D' = (e - e')v \bmod n.$$
Since $(e - e')$ is invertible modulo n we deduce that
$$p^{2j_1} + \cdots + p^{2j_N} = v \bmod n.$$
In other words, the witness (v, r_y) consists of a correctly formed vote on the form $v = M^{j_1} + \cdots + M^{j_N}$, where $0 \leq j_1 < \cdots < j_N < L$, and the random number r_y involved in the encryption. This shows that the protocol satisfies the special soundness property.

Special honest verifier zero-knowledge. Given the common input and a challenge $e \in \{0, \ldots, 2^t - 1\}$ we wish to simulate a proof of the encryption containing a vote on the right form.

We start by picking random $r_1, \ldots r_N, r'_1, \ldots, r'_N \in \mathcal{R}_K$ and form commitments $c_1 = com_K(p^0; r_1), \ldots, com_K(p^{N-1}; r_N)$ and $c'_1 = com_K(1; r'_1), \ldots, c'_N = com_K(1; r'_N)$. Note that by the hiding property of the commitment scheme these commitments are indistinguishable from properly formed commitments to $p^{j_1}, \ldots, p^{j_N}$ and $p^{j_2 - j_1 - 1}, \ldots, p^{L - j_N - 1}$.

We now pick $f_1, \ldots, f_N$ as shadows for $ep^{j_1}, \ldots, ep^{j_N}$ and D as a shadow for $f_1 p^{j_1} + \ldots f_N p^{j_N}$. With this choice of $f_1, \ldots, f_N, D$ they are indistinguishable from the $f_1, \ldots f_N$ and D of a real proof. by the definition of shadows.

We may also pick $z_{1,1}, \ldots, z_{1,N}, z_{2,1}, \ldots z_{2,N}, z_3 \in \mathcal{R}_K$ and $z_4 \in \mathcal{R}_{pk}$ as random shadows so that thy are indistinguishable from those in a real proof.

We can now compute $y_\gamma = E_{pk}(D; z_4)y^{-e}$, $c_\gamma = c_1^{-f_1} \cdots c_N^{-f_N} com_K(D; z_3)$, $c_{d_1} = com_K(f_1; z_{1,1}c_1^{-e}, \ldots, c_{d_N} = com_K(f_N; z_{1,N})c_N^{-e})$ and $c_{d_1 b} = com_K(0; z_{2_1})^{-1} c_1^{pf_1} c_2^{-e}, \ldots,$ $c_{d_N b} = com_K(0; z_{2,N})^{-1} c_N^{pf_n} c_{N+1}^{-e}$.

With these choices, we have a simulated proof that due to the hiding property of the commitment scheme and the semantic security of the cryptosystem looks entirely like a normal proof with challenge e. This means that we have demonstrated the special honest verifier zero-knowledge property of the proof system.

Finally. we see from the proof of special honest verifier zero-knowledge that if the commitments $c_1, \dots, c_N, c'_1, \dots, c'_N$ are statistically hiding and that all the shadows and random shadows are statistically hiding, then the entire proof system is statistical special honest verifier zero-knowledge.□

If we want, we can make some small alternations to the protocol above to open for the possibility for a voter to vote on less then N candidates and/or the possibility for a voter to vote multiple times on the same candidate. The possibility for a voter to vote on less then N candidates can be obtained by including N dummy candidates, while if we want the voter to be able to vote multiple times on the same candidate, we can let $c'_1, \dots c'_N$ instead be commitments to $p^{j_2-j_1}, \dots, p^{L-j_N}$ and, at the same time, remove the power p in the multiplication proof $(c_i)(c'_i)^p = c_{i+1}$.

We will now present a randomized verification algorithm which makes the verification procedure in Σ-protocol above more efficient Notice that one thing we must do several times during the verification procedure is to compute two elements in $\mathcal{C}_K$ in two different ways, such as $com_K(f; z_1)$ and ac^e, and then check to see if they are identical. Since computing some of these elements can be complicated we want to reduce the time used in this process.

As an example assume we are given multiple pairs $(c_1, d_1, \dots c_n, d_n)$ in $\mathcal{C}_K$ and that we wish to check if these are pairwise identical. To do this we can choose $s_1, \dots, s_n$ at random from $[0, \dots, 2^t - 1]$ and then check if $c_1^{s_1} \cdots c_n^{s_n}$ equals $d_1^{s_1} \cdots d_n^{s_n}$. The reason we can do this is, given that $\mathcal{C}_K$ is a group with no non-trivial elements of order less then 2^t, we have with probability at least $1 - 2^{1-t}$ that $c_1^{s_1} \cdots c_n^{s_n} \neq d_1^{s_1} \cdots d_n^{s_n}$ if there exists i such that $c_i \neq d_i$ [DGS03]. Here t may be a smaller security parameter than in the Σ-protocols since the computation happens only on the verifiers side, and the prover has therefore no opportunity to actively cheat.

The reason this is interesting is that in the Σ-protocol $c_i = com_K(m_i; r_i)$ for some m_i, r_i known to the verifier. This means that $c_1^{s_1} \cdots c_n^{s_n}$ can be computed as $com_K(s_1 m_1 + \cdots + s_n m_n; r_1 s_1 + \cdots + r_n s_n)$. Given that the binary operations are faster to compute in $\mathcal{M}_K$ and $\mathcal{R}_K$ than in $\mathcal{C}_K$ this will make to verification more efficient. Since the probability of catching any cheating grows exponentially with t, we can typically choose t reasonably small. Accordingly, the extra computational effort required to compute the additional exponentiations to $s_1, \dots, s_n$ is dwarfed by the savings we get by not having to verify each commitment opening itself [DGS03].

Using this idea in the voting scheme on multiple candidates the verification procedure, after some calculations becomes the following.

Randomized verification algorithm

Verification. Check that $c_{d_1}, \dots, c_{d_n}, c_{d_1 b}, \dots, c_{d_N b}, c_\gamma \in \mathcal{C}_K,\ y_\gamma \in \mathcal{C}_{pk},\ f_1, \dots, f_N, D \in \mathbb{Z}$, $z_{1,1}, \dots, z_{1,N}, z_{2,1}, \dots, z_{2,N}, z_3 \in \mathcal{R}_K$ and $z_4 \in \mathcal{R}_{pk}$.
Select at random $s_1, \dots, s_N, s'_1, \dots, s'_N, s \in \{0, \dots, 2^t - 1\}$. Verify that

$$com_K(s_1 f_1 + \cdots + s_N f_N + sD + es'_N p^L; s_1 z_{1,1} + \cdots + s_N z_{1,N} + s'_1 z_{2,1} + \cdots + s'_N z_{2,N} + z_3)$$
$$= c_\gamma^s c_{d_1}^{s_1} \cdots c_{d_N}^{s_N} (c_{d_{1b}}^{-1})^{s'_1} \cdots (c_{d_{Nb}}^{-1})^{s'_N} c_1^{es_1 + sf_1} c_2^{e(s_2 - s'_1) + sf_2} \cdots$$
$$c_N^{e(s_N - s'_{N-1}) + sf_N} (c'_1)^{pf_1 s'_1} \cdots (c'_N)^{pf_N s'_N}$$

3.4 The voting system

We are now ready to present the voting system. As before, let $M = p^2$, where p is a prime, be a strict upper bound on the number of voters, let L be the number of candidaties, and let N be the total number of candidates a voter may vote on. A vote on candidates number $0 \leq j_1 < \dots j_N < L$ is represented as the number $M^{j_1} + \cdots + M^{j_N}$.

The electronic voting system based on homomorphic encryption.

Set-up. The election authorities generates the public-key pk of the encryption scheme in Section 3.2.2, and distributes the private key x amongst themselves. Also the election authorities generates the public-key K of the Damgård-Fujisaki commitment scheme in Section 3.2.1.

Voting phase. Each voter i encrypts his vote $y_i = E_{pk}(M^{j_1} + \cdots + M^{j_N}; r_i)$ and uses the Σ-protocol of Section 3.3 to prove that y_i indeed is a correctly formed vote.

Tallying. The voting authorities computes

$$y = \prod_{i \in A} y_i = E_{pk}(v_0 M^0 + \cdots + v_{L-1} M^{L-1}; r),$$

where A is the collection of accepted votes, v_j is the number of votes on candidate j and $r = \sum_{i \in A} r_i$. Finally the election authorities jointly decrypt y using the threshold decryption described in Section 3.2.2 to reveal the results of the election.

Theorem 3.6. Under the discrete logarithm assumption, and the RSA assumption the voting system above satisfies universal verifiability, computational robustness and computational privacy.

Proof. First consider privacy. Theorem 3.2 ensures that no information about an encrypted vote leaks under generalized the Decision Diffie Hellman assumption (which is at least as strong as the discrete logarithm assumption). By Theorem 3.5 and Proposition 2.1 the Σ-protocol do not reveal any information about the vote if the commitment scheme, shadows and random shadows are statistically hiding. We know that commitments are statistically hiding from Theorem 3.1. If the Σ-protocol is done non-interactively using Fiat-Shamir heuristics the protocol is zero-knowledge by the discussion in the Section 2.3.3. Finally by Theorem 3.3 privacy is preserved under the discrete logarithm assumption as long as at most $t-1$ authorities collude or are hacked into.

As for robustness, we note that in order for a voter to cast a non-valid vote, he must somehow during the Σ-protocol convince the authorities that the commitment, and therefore the encryption containing the non-valid vote, is of the correct form. Assuming that the voter can not break the binding property of the commitment scheme, Theorem 2.1 shows that this can only happen with probability 2^{-k}, where k is the length of the bit-string challenge used in the Σ-protocol. Thus, in order for a voter to cast a non-valid vote with probability greater then 2^{-k} the voter must break the binding property of the commitment scheme which, by Theorem 3.1, is at least as hard as solving the RSA problem. On the other hand, if some of the authorities try to cheat Theorem 3.3 ensures that robustness is preserved as long as some threshold t of the authorities is operating properly.

Finally, universal verifiability is satisfied since a passive observer can check that the product of the votes is equal to the product computed by the authorities. Moreover, the passive observer can verify that the final tally was decrypted using only values from authorities who passed the proof of knowledge, thereby ensuring that the correct shares was used during decryption. $\square$

4

An electronic voting system based on verifiable secret sharing

4.1 Introduction

We will now present an electronic voting system based on verifiable secret sharing. The voting system we present is from "Multi-authority secret-ballot election with linear work" by Ronald Cramer, Matthew Franklin, Barry Schoenmakers and Moti Yung [CFSY96].

We assume two basic means of communication is available to the parties in the system. The first is a *bulletin board*, which is a broadcast channel with memory that can be observed and read by all parties. Each party controls his own section of the board in the sense that he can post messages exclusively to his own section, but not to the extent that he can erase or overwrite previously posted messages. The second mean of communication are *private channels* to support private communication between the voters and authorities. For this task any secure public-key encryption scheme is suitable, possibly using the bulletin board to post corresponding encryptions.

The participants in the scheme are n authorities $A_1, \dots A_n$ and m voters $(V_1, \dots V_m)$. Informally the scheme work as follows. Each voter V_i prepares a vote by choosing a random $b_i \in \{1, -1\}$ and encrypts it resulting in B_{i0}. After preparing the vote V_i publishes B_{i0} to the bulletin board. Then each voter acts as the dealer in a verifiable secret sharing scheme and distributes a verifiable share of b_i to each authority using a private channel. The voter also post some values proof(B_{i0}) to the bulletin board to prove that B_{i0} indeed encrypts a value in $\{1, -1\}$. Later, voter V_i, may cast a vote $v_i \in \{1, -1\}$ by publishing a value $d_i = b_i v_i$. At the end of the election the authorities can compute the aggregate value $T = \sum_{i=1}^{m} v_i$, which represents the total number of yes-votes minus the total number of no-votes. How this value can be computed from the shares will be explained in Section 4.3.3.

In the next section we will present the encryption scheme together with the commitment scheme needed for the verifiable secret sharing scheme, which we present in Section 4.3. Finally the voting system is presented in Section 4.4.

4.2 The commitment and encryption scheme

As mentioned we will be using verifiable secret sharing (VSS) to ensure privacy in our election scheme. Unfortunately, most VSS schemes, such as [CDM00], relay on exponentially many interactions between the shareholders, which would be to time consuming for a big-scale election. By the results of [Ped92] no non-interactive VSS scheme which does not relay on some computational problem exists. We will follow [Ped92] who uses a commitment scheme based on the assumption that the discrete logarithm problem is intractable to construct a non-interactive verifiable secret sharing scheme. Remember that the discrete logarithm assumption is the assumption that given a group G, an element $g \in G$ of order n and an element $h \in \langle g \rangle$, the problem of finding an unique integer a such that $g^a = h$ is intractable.

Let $\mathcal{G} = \{\mathcal{G}_k\}$ be a family of groups of prime order such that the group operations can be performed efficiently, group elements can be efficiently sampled with uniform distribution and group membership as well as equality of group elements can be efficiently tested. Let *Gen* be a probabilistic polynomial time generator that on input 1^k outputs a description of a group $G \in \mathcal{G}_k$

Voter			Verifier
$v = 1$	$v = -1$		
$\alpha, z_{-1}, e_{-1}, r_1 \in_R \mathbb{Z}_q$	$\alpha, z_1, e_1, r_{-1} \in_R \mathbb{Z}_q$		
$B \leftarrow g^{\alpha} h$	$B \leftarrow g^{\alpha}/h$		
$a_{-1} \leftarrow g^{z_{-1}}(Bh)^{-e_{-1}}$	$a_{-1} \leftarrow g^{r_{-1}}$		
$a_1 \leftarrow g^{r_1}$	$a_1 \leftarrow g^{z_1}(B/h)^{-e_1}$	$\xrightarrow{B,a_{-1},a_1}$	
			$s \in_R \mathbb{Z}_q$
		$\xleftarrow{s}$	
$e_1 \leftarrow s - e_{-1}$	$e_{-1} \leftarrow s - e_1$		
$z_1 \leftarrow r_1 + \alpha e_1$	$z_{-1} \leftarrow r_{-1} + \alpha e_{-1}$		
		$\xrightarrow{e_{-1},e_1,z_{-1},z_1}$	
			$s \overset{?}{=} e_{-1} + e_1$
			$g^{z_{-1}} \overset{?}{=} a_{-1}(Bh)^{e_{-1}}$
			$g^{z_1} \overset{?}{=} a_1(B/h)^{e_1}$

Figure 4.1: Encryption and proof of validity of ballot B. Here $\in_R \mathbb{Z}_q$ means that the elements are chosen at random from $\mathbb{Z}_q$, while $\overset{?}{=}$ means that the verifier checks if the equality holds.

(including the group order) and two random elements g, h from G. Each family $\mathcal{G}$ for which it is reasonable to assume the intractability of the discrete logarithm problem is suitable for our purpose of constructing an efficient and secure commitment scheme. A well-known family however, is obtained by choosing large primes p and q at random such that q divides $p - 1$. Then G is the unique subgroup of order q in $\mathbb{Z}_p^*$.

We will now present Pedersens commitment scheme together with an efficient proof of validity from [CFSY96]. The reason we include a proof of validity is that, in the election scheme to follow, we will use the commitment scheme together with the proof of validity to encrypt a value $v \in \{1, -1\}$. For later we will refer to the below as a commitment scheme if we do not use the proof of validity, and as an encryption scheme if we do[1].

The Pedersen commitment scheme with efficient proof of validity.

Set-up. The participants, or a designated subset of them, run $Gen(1^k)$ and obtain a group G_q of prime order q, and random group elements g and h.

Commit. A commitment to $v \in \mathbb{Z}_q$ is formed by choosing $\alpha \in \mathbb{Z}_q$ at random and letting the commitment be $B = com_{(G_q,g,h)}(v, \alpha) = g^{\alpha} h^{v}$.

Open. A participant can later open B by revealing both v and α. A verifying party then checks whether $B = g^{\alpha} h^{v}$, and accepts v as the committed value.

Proof of validity. To demonstrate that an encrypted value v is indeed in $\{1, -1\}$, without revealing it, the voter and the verifier execute the efficient proof of validity in Figure 4.1.

The commitment and encryption scheme satisfy the following property which we will note for later.

[1] The word encryption here is a bit misleading since the "encrypted" value never will (or can) be decrypted. We will use the encryption in the voting system in the same way as the commitments will be used in the upcoming verifiable secret sharing scheme. All this aside, we will still follow [CFSY96] and call this part an encryption.

Homomorphic property. The commitment and encryption scheme is homomorphic, that is for all $\alpha, v \in \mathbb{Z}_q$

$$com_{(G_q,g,h)}(v_1,\alpha_1)com_{(G_q,g,h)}(v_2,\alpha_2) = com_{(G_q,g,h)}(v_1 + v_2 \bmod q, \alpha_1 + \alpha_2 \bmod q)$$

Theorem 4.1. Under the discrete logarithm assumption, the commitment and encryption scheme is unconditionally hiding and computationally binding. Furthermore, the proof of validity is a witness indistinguishable Σ-protocol proving that a given encryption is indeed an encryption of a value from the set $\{1, -1\}$, thereby releasing no information about the actual value.

Proof. To prove unconditionally hiding note that for every fixed v the map $com_{G_q,g,h}(v,\alpha) : \mathbb{Z}_q \to G_q$ is an isomorphism. It follows that the distribution of $com_{G_q,g,h}(v,\alpha)$ is uniform in G_q.

If any party is able to open a commitment or encryption c in two different ways, i.e to present values α, v, α, v' such that $B = g^{\alpha}h^{v} = g^{\alpha'}h^{v'}$ with $\alpha \neq \alpha'$ and $v \neq v'$, it follows that they can compute $\log_g h = \frac{\alpha-\alpha'}{v'-v}$, which contradicts the discrete logarithm assumption.

To prove that the proof of validity is a witness indistinguishable Σ-protocol for the relation,

$$R_{OR} = \{(v,\alpha) | B = g^{\alpha}h^{v}, v = 1 \text{ or } v = -1\},$$

we will construct the protocol using the protocol $\mathcal{P}_{OR}$ from Section 2.3.2. The proof then follows from Theorem 2.3. As the sub-protocol $\mathcal{P}$ of the protocol $\mathcal{P}_{OR}$ we can use a variant of Schnorr's protocol for proving knowledge of discrete logarithms from the example in Section 2.3. Remember that Schnorr's protocol, proving that we know $\alpha = \log_g h$ in a group of prime order q, is of the following form. The prover sends $a = g^r$ and answers the random challenge e with $z = r + e\alpha$. The verifier then checks that $g^z = ah^e$.

We now want to prove that we know $\alpha = \log_g(Bh^{-v}), v \in \{1, -1\}$ so we need to do a minor adjustments to the protocol. Consider the following: The prover sends $a = g^r$ and answers the random challenge e with $z = r + e\alpha$. The verifier then checks that $g^z = a(Bh^{-v})^e$. Since $g^z = g^{r+e\alpha} = g^r(g^{\alpha})^e = a(bh^{-v})^e$ completness trivially holds with probability 1. Special soundness holds by the same argument as Schnorrs original scheme. That is, answer to two different challenges gives the two equations $z_1 = r + e_1\alpha \bmod q$ and $z_2 = r + e_2\alpha \bmod q$, which combined yields $\alpha = (z_1 - z_2)(e_1 - e_2)^{-1} \bmod q$. To simulate an accepting conversation we just choose $z, e \in \mathbb{Z}_q$ at random and compute $a = g^z(Bh^{-v})^{-e}$.

Below we use the above protocol as the subprotocol $\mathcal{P}$ in the protocol $\mathcal{P}_{OR}$ from Section 2.3.2 to obtain a Σ-protocol for the relation R_{OR}. The "blueprint" for construction is written in *italic*, while $\Rightarrow$ and $\Leftarrow$ represents what the prover should do if $v = 1$ and $v = -1$ and respectivly. If the action is the same in both cases we represent this by $\Leftrightarrow$.

1. *P computes the first message a_b in $\mathcal{P}$ using x_b, w as input.*

   $\Rightarrow$ P chooses $r_1 \in \mathbb{Z}_q$ at random and computes $a_1 = g^{r_1}$.

   $\Leftarrow$ P chooses $r_{-1} \in \mathbb{Z}_q$ at random and computes $a_{-1} = g^{r_{-1}}$.

   P chooses e_{1-b} at random and runs the simulator M on input x_{1-b}, e_{1-b}, let $(a_{1-b}, e_{1-b}, z_{1-b})$ be the output.

   $\Rightarrow$ P chooses $z_{-1}, e_{-1} \in \mathbb{Z}_q$ at random and computes $a_{-1} = g^{z_{-1}}(Bh)^{-e_{-1}}$.

   $\Leftarrow$ P chooses $z_1, e_1 \in \mathbb{Z}_q$ at random and computes $a_1 = g^{z_1}(B/h)^{-e_1}$.

   P sends a_0, a_1 to V.

   $\Leftrightarrow$ P sends a_{-1}, a_1 to V.

2. *V chooses a random t-bit string s and sends it to P.*

   $\Leftrightarrow$ V chooses at random $s \in \mathbb{Z}_q$ and sends it to P.

3. *P sets $e_b = s \oplus e_{1-b}$ and computes the answer z_b in $\mathcal{P}$ to challenge e_b using x_b, a_b, e_b, w as input. He sends e_0, z_0, e_1, z_1 to V.*

$\Rightarrow$ P sets $e_1 = s - e_{-1}$ and computes $z_1 = r_1 + e_1\alpha$.

$\Leftarrow$ P sets $e_{-1} = s - e_1$ and computes $z_{-1} = r_{-1} + e_{-1}\alpha$

$\Leftrightarrow$ P sends e_{-1}, z_{-1}, e_1, z_1 to V.

V decides to accept the proof iff $s = e_0 \oplus e_1$ and both (a_0, e_0, z_0) and (a_1, e_1, z_1) are accepting conversations.

$\Leftrightarrow$ V verifies that $s = e_1 + e_{-1}$, $g^{z_1} = a_1(B/h)^{e_1}$, $g^{z_{-1}} = a_{-1}(Bh)^{e_{-1}}$.

It is easy to see that the protocol above is the same as the proof of validity in Figure 4.1. By Theorem 2.3 the proof of validity in Figure 4.1 is therefore a witness indistinguishable Σ-protocol proving that a given encryption is indeed an encryption of a value from the set $\{1, -1\}$. □

Jumping ahead a bit, envision that the voter posts an encryption of his vote and that all other participants must verify its validity. As depicted in Figure 4.1, a source of randomness is required in the program for the verifier. For this purpose, one can use some unpredictable physical source of randomness [CF85], or agree on mutually random bits by cryptographic means. A more practical way is of course to use the Fiat-Shamir heurstics discussed in Section 2.3.3 [CFSY96]. Let $\mathcal{H}$ be a suitible strong cryptographic hash function thought of as a random oracle. In the non-interactive version of our proof of validity, the challenge s is computed as $s = \mathcal{H}(b, a_{-1}, a_1)$. The set of values e_{-1}, z_{-1}, e_1, z_1 is denoted by proof(B). Given the values in proof(B), any participants can check the validity of B by verifying that $e_{-1} + e_1 = \mathcal{H}(B, g^{z_{-1}}(Bh)^{-e_{-1}}, g^{z_1}(B/h)^{-e_1})$. Similar as in Section 3.3, a bit string specific to each voter can be included in the hash function to prevent vote duplication. In case the proof is done interactivly this bit string could be incorporated in the challange.

4.3 Pedersens verifiable secret sharing scheme

4.3.1 Introduction

We will now present Pedersens verifiable secret sharing scheme, which is an extension of Shamirs scheme from Section 2.4. Let q be a prime. Rembember that in Shamirs scheme the dealer, D, distributes a secret s by choosing a random polynomial $f \in \mathbb{Z}_q[x]$, of degree at most $t-1$ such that $f(0) = s$ and then give each player P_i the share $f(i)$. A set S of t or more players can later find s from the formula

$$s = \sum_{i \in S} s_i \prod_{j \in S \setminus \{i\}} \frac{-j}{i-j}.$$

Our goal is to extend this scheme with a verification protocol, VP, such that any participant who have (honestly) accepted their shares in VP can find s. More formally VP must satisfy the following definition from [Ped92]. Here $|x|$ is defined as the length of the binary representation of an integer x.

Definition. A *verification protocol*, VP, takes place between D and $P_1, \ldots, P_n$. It must satisfy the following two requirements.

1. If the dealer follows the distribution protocol and if both the dealer and P_i follow VP, then P_i accepts with probability 1.

2. For all subsets S_1 and S_2 of $\{1, \ldots, n\}$ of size t such that all parties $(P_i)_{i \in S_1}$ and $(P_i)_{i \in S_2}$ have accepted their shares in VP the following holds except with negligible probability in $|q|$: If s_i is the secret computed by the participants in S_i (for $i = 1, 2$) then $s_1 = s_2$.

A share is *correct* if it is accepted in VP.

Even though the definition above allows for any kind of interaction between the participants we will be concerned with non-interactive verification protocols. In the non-interactive version the dealer distributes some extra information along with the shares, and in the verification protocol each player verify that their share is consistent with this extra information.

Note that, in the definition above, there is no reference to the actual secret s the dealer is distributing when defining the correctness of a share. The reason for this is that during the verification protocol the players have no information about s and thus the secret is whatever the dealer claims. After the verification protocol the secret is defined as the value which any t participants will find when combining their shares. This is not well defined if the dealer manages to distribute inconsistent shares[2], but the definition guarantees that the dealer will almost always get caught when trying to cheat.

4.3.2 The scheme

Let a group G_q and two random elements g, h be given such that Pedersens commitment scheme from Section 4.2 can be applied, and let $K = (G_q, g, h)$ The following is then a non-interactive verifiable secret sharing scheme found in [Ped92].

Pedersen's verifiable secret sharing scheme.

1. D publishes a commitment, using Pedersen's commitment schmeme, to s: $B_0 = com_K(s, \alpha)$ for a randomly chosen $\alpha \in \mathbb{Z}_q$.
2. D chooses $H \in \mathbb{Z}_q[x]$ of degree at most $t-1$ satisfying $H(0) = s$ at random, and computes $s_i = H(i)$ for $i = 1, \dots, n$.

   Let $H(x) = s + H_1 x + \cdots + H_{t-1}x^{t-1}$. D chooses $G_1, \dots, G_{t-1} \in \mathbb{Z}_q$ at random and uses G_i when committing to F_i for $i = 1, \dots, t-1$. D broadcast $B_i = com_K(H_i, G_i)$ for $i = 1, \dots, t-1$.
3. Let $G(x) = \alpha + G_1 x + \cdots + G_{t-1}x^{t-1}$ and let $a_i = G(i)$ for $i = 1, \dots, n$. Then D sends (s_i, a_i) to each player P_i via a private channel.

When P_i has received his share (s_i, a_i) he verifies that

$$com_K(s_i, a_i) = \prod_{l=0}^{t-1} B_l^{i^l}. \tag{4.1}$$

Lemma 4.1. Let $S \subset \{1, \dots, n\}$ be a set of t players such that equation 4.1 holds for these t players. Then these players can find a pair (s', a') such that $B_0 = g^{a'}h^{s'}$.

Proof. Using Lagrange interpolation the players in S can find two unique polynomials H', G' satisfying $H'(i) = s_i$ and $G'(i) = a_i$ for all $i \in S$. Let $h = g^d$. Then

$$h^{H'(i)+dG'(i)} = com_K(s_i, a_i) = h^{s_i + da_i},$$

for all $i \in S$. Thus $(H' + dG')(i)$ is the unique polynomial of degree at most $t-1$ mapping i to $s_i + da_i$. Let $B_j = g^{G_j}h^{H_j} = h^{H_j + dG_j} = h^{b_j}$, then

$$b(x) = \sum_{j=0}^{t-1} b_j x^j = \sum_{j=0}^{t-1}(H_j + dG_j)x^j = \sum_{j=0}^{t-1} H_j x^j + d\sum_{j=0}^{t-1} G_j x^j,$$

[2]Shares that not all lie on the same polynomial of degree at most $t-1$.

satisfies $b(i) = s_i + da_i$ for all $i \in S$. And in particular

$$B_0 = h^{b_0} = h^{H'(0)+dG'(0)} = g^{G'(0)}h^{H'(0)}.$$

Therefore it is sufficient to put $s' = H'(0)$ and $a' = G'(0)$. $\square$

Note that the players in S do not have to find H' in order to find the secret. It is more efficient to use the formula $s = \sum_{i\in S} \lambda_{S,i}s_i$, where $\lambda_{S,i} = \prod_{j\in S\setminus\{i\}} \frac{i}{i-j}$. We can also find α by the formula $\alpha = \sum_{i\in S} \lambda_{S,i}a_i$.

Theorem 4.2. Under the assumption that the dealer cannot find $\log_g h$ except with negligible probability in $|q|$, equation 4.1 satisfy the definition of a verification protocol.

Proof. If all participants follow the protocol, then P_i always accepts since

$$\prod_{l=0}^{t-1} B_l^{i^l} = \prod_{l=0}^{t-1}(g^{G_i}h^{H_i})^{i^l} = g^{\alpha+G_1 i+\cdots+G_{t-1}i^{t-1}}h^{s+H_1 i+\cdots+H_{t-1}i^{t-1}} = g^{a_i}h^{s_i} = com_K(s_i;a_i).$$

Now let D^* be an algorithm that on input (s,n,t) outputs two subsets S_1, S_2 of $(1,\ldots,n)$ of size t and $(B_0,\ldots B_{t-1},(s_i,a_i)_{i\in S_1},(s_j,a_j)_{j\in S_2})$ such that for all $i \in S_1$ and all $i \in S_2$,

$$com_K(s_i,a_i) = \prod_{l=0}^{t-1} B_l^{i^l}.$$

According to Lemma 4.1 we can use the sets S_1 and S_2 to find pairs (s_1,a_1) and (s_2,a_2) respectively such that $B_0 = com_K(s_1,a_1) = com_K(s_2,a_2)$. Now assume that D^* has output shares such that $s_1 \neq s_2$. Then by the proof of Theorem 4.1 we can find $\log_g h$ as $\frac{a_1-a_2}{s_2-s_1}$. This shows that computing $\log_g h$ is at least as hard as distributing inconsistent shares, and the theorem follows. $\square$

As a consequence of Theorem 4.2 all the shares satisfying the verification procedure are consistent unless the dealer succeeds in finding $\log_g h$ *before* the last share has been sent.

The following theorem shows that fewer than t players get no information about the secret. For any subset $S \subset \{1,\ldots,n\}$, let $views_S$ denote all the messages the players in S see,

$$views_S = (B_0,\ldots,B_{t-1},(s_i,a_i)_{i\in S})$$

Theorem 4.3. For any $S \subset \{1,\ldots,n\}$ of size at most $t-1$, Prob[D has secret $s|views_S$] =Prob[D has secret s] for all $s \in \mathbb{Z}_q$.

Proof. It is sufficient to show that any set of $t-1$ players get no information about the secret. The theorem will then follow by the fact that any set of less then $t-1$ players holds even less information.

So let $S \subset \{1,\ldots,n\}$ be a set of $t-1$ players, let $views_S = \{B_0,\ldots,B_{t-1},(s_i,a_i)_{i=1,\ldots,t-1}\}$, and let (s,α) be the values distributed by the dealer using polynomials $H(x) = s+H_1x+\cdots+H_{t-1}(x)$ and $G(x) = \alpha + G_1x + \cdots + G_{t-1}x^{t-1}$. For any $s' \in \mathbb{Z}_q$, there exists a unique $\alpha' \in \mathbb{Z}_q$ such that $com_K(s',\alpha') = B_0$. And, by Theorem 2.4, for every such pair (s',α') there exists unique polynomials H' and G' of degree at most $t-1$ satisfying

$$H'(0) = s', H'(i) = s_i, i = 1,\ldots,t-1,$$

$$G'(0) = \alpha', G'(i) = a_i, i = 1,\ldots,t-1.$$

Let $H'(x) = s' + H_1'x + \cdots + H_{k-1}'x^{k-1}$ and $G'(x) = \alpha' + G_1'x + \cdots + G_{t-1}'x^{t-1}$. In order to show that $views_S$ contains no information about the secret, we must show that $B_i = com_K(H_i', G_i')$ for $i = 1,\ldots,t-1$, since this is true for the polynomials chosen by the dealer. As in the proof of Lemma 4.1 there is one and only one polynomial of degree at most $t-1$ satisfying $h^{f(i)} = g^{a_i}h^{s_i}$ (with $s_0 = s, a_0 = \alpha'$), and $(H' + dG')(x)$ satisfies this for d =$\log_g h$. But since $H(i) = H'(i)$ and

$G(i) = G'(i)$ for $i = 1, \ldots t-1$ and for $i = 0$ we have $h^{s+d\alpha} = g^{\alpha}h^{s} = B_0 = g^{\alpha'}h^{s'} = h^{s'+d\alpha'}$, it follows that $(H + dG)(x)$ satisfies this aswell. This means that for each $i = 1, \ldots, t-1$ we must have $H_i + dG_i = H'_i + dG'_i$, which gives

$$com_K(H_i, G_i) = g^{G_i}h^{H_i} = h^{H_i+dG_i} = h^{H'_i+dG'_i} = g^{G'_i}h^{H'_i} = com_K(H'_i, G'_i)$$

This means that for the players in S, each $s' \in \mathbb{Z}_q$ is consistent with the information in $view_S$, and the theorem follows. □

By the results above we see that Pedersen's scheme satisfies the definition of verifiable secret sharing from Section 2.4. If the dealer is honest, then all sets of t honest participants will be able to reconstruct the secret by Theorem 4.2. No corrupted player can disrupt the reconstruction since all participants can verify that the shares posted by the players during reconstruction satisfy equation 4.1. This means that the reconstruction will be a success as long as at most $n - t$ players are corrupted. If the dealer is corrupt, any player receiving inconsistent shares can post a complaint. The dealer must then publish the share sent to that player. If there are $n - t$ or more complaints the dealer is deemed corrupt, while if the distribution is accepted t or more players can reconstruct the secret by Theorem 4.2.

4.3.3 Linear combination of shares

Assume that two secrets s' and s'' has been distributed by Pedersens scheme. Because of the homomorphic property of the commitments and the linear property of polynomials[3], each player can locally compute on their shares in s', s'' to get shares in any linear combination of s' and s''. To see this let (s'_i, a'_i) and (s''_i, a''_i) be P_i's share of s' and s'' respectively, and let $(B'_0, \ldots, B'_{t-1})$ and $(B''_0, \ldots, B''_{t-1})$ be the broadcasted messages when the two secrets where distributed. Each player P_i can then compute $(B_0, \ldots, B_{t-1})$ corresponding to a verifiable distribution of $s = s' + s''$ mod q by the formula $B_j = B'_j B''_j$ for $j = 0, \ldots, t-1$. Furthermore P_i's secret share, (s_i, a_i) of s is given by $s_i = s'_i + s''_i$ mod q and $a_i = a'_i + a''_i$ mod q. By insertion it is easy to see that if both (s'_i, a'_i) and (s''_i, a''_i) are correct shares, then (s_i, t_i) is a correct share of s,

$$g^{a_i}h^{s_i} = g^{a'_i+a''_i}h^{s'_i+s''_i} = g^{a'_i}h^{s'_i}g^{a''_i}h^{s''_i} = \prod_{l=0}^{t-1}(B'_l)^{i^l}\prod_{l=0}^{t-1}(B''_l)^{i^l} = \prod_{l=0}^{t-1}(B'_lB''_l)^{i^l} = \prod_{l=0}^{t-1}B_l^{i^l}.$$

If we instead want to compute the secret as $s = ks'$ mod q for some $k \in \mathbb{Z}_q$, then P_i can compute his share (s_i, a_i) and $(B_0, \ldots, B_{t-1})$ as follows, $s_i = ks'_i$ mod q, $a_i = ka'_i$ mod q and $B_j = (B'_j)^k$ for $j = 0, \ldots, t-1$. Again we see that (s_i, a_i) is a correct share by insertion,

$$g^{a_i}h^{s_i} = g^{ka'_i}h^{ks'_i} = (g^{a'_i}h^{s_i})^k = (\prod_{l=0}^{t-1}(B'_l)^{i^l})^k = \prod_{l=0}^{t-1}(B'^k_l)^{i^l} = \prod_{l=0}^{t-1}(B_l)^{i^l}.$$

In both the above cases Lemma 4.1 implies that any t shareholders who have accepted their shares in s' and s'' can find a pair (s, a) such that $g^a h^s = E_0$. Furthermore, it is an immediate consequence of Theorem 4.3 that fewer than t persons have no information about s if s' and s'' is distributed correctly.

4.4 The voting system

We are now ready to present the voting system of [CFSY96]. Again, the participants in the scheme are n authorities $A_1, \ldots A_n$ and m voters $(V_1, \ldots V_m)$. Privacy and robustness of the scheme are as follows. No collusion of fewer then t authorities can reveal an individual vote, while the election is successful if at least t authorities operate properly $(1 \leq t \leq n)$. We include a mechanism

[3]That is given $H(x)$ and $H'(x)$, then $aH(x) + bH'(x) = (aH + bH')(x)$

to postpone the decision on what to vote until the preparation of the election is complete. This enables most of the work to be done offline before the election, while voting is simplified to posting just one bit of information.

The voting system work as follows. Each voter V_i prepares a vote by choosing a random $b_i \in \{1, -1\}$ and encrypt it as $B_{i0} = g^{\alpha} h^{b_i}$ using a random $\alpha \in \mathbb{Z}_q$, and posts B_{i0} to the bulletin board. Then each voter acts as the dealer in Pedersens scheme with threshold t and distributes a verifiable share (b_{ij}, a_{ij}) of b_i to authority A_j via a private channel, and publishes the commitments $(B_{i1}, \dots B_{i,t-1})$. The voter also post proof(B_{i0}) to the bulletin board to prove that B_{i0} indeed encrypts a value in $\{1, -1\}$. Later, voter V_i, may cast a vote $v_i \in \{1, -1\}$ by publishing a value $d_i = b_i v_i$. Now each authority A_j can compute their shares (T_j, S_j) in the final results in the manner explained in Section 4.3.3. The share $(d_i b_{ij}, d_i a_{ij})$ is A_j's share in v_i, and thus $(T_j, S_j) = (\sum_{i=1}^{n} d_i b_{ij}, \sum_{i=1}^{n} d_i a_{ij})$ is a share in the final results. Now, by the discussion of the last section, $(B_{il}^{d_i})_{l=0,\dots t-1}$ corresponds to a verifiable distribution of $v_i = d_i b_i$, and furthermore $(\prod_{i=0}^{m} B_{il}^{d_i})_{l=0,\dots,t-1}$ corresponds to a verifiable distribution of (T_j, S_j). So, when A_j publishes his share (T_j, S_j) during reconstruction of the final tally, each authority can check that this value is correct by verifying that

$$g^{S_j} h^{T_j} = \prod_{l=0}^{t-1} (\prod_{i=0}^{m} B_{il}^{d_i})^{j^l}$$

At the end of the election t authorities with correct shares can reconstruct the aggregate value $T = \sum_{i=1}^{m} v_i$ reduced modulo q such that $-q/2 < T < q/2$ represents the total number of yes-votes minus the total number of no-votes. The total number of yes-votes is thus $(m + T)/2$. For these numbers to be correct it is a requirement that $m < q/2$.

We assume that the group G_q and the members g and h are generated as described in Section 4.2. In particular, it then follows that $\log_g h$ is not known to any participants.

Ballot construction

Each voter V_i prepares a masked vote $b_i \in \{-1, 1\}$ in the following way.

1. The voter chooses b_i randomly from $\{-1, 1\}$ and computes the ballot $B_{i0} = g^{\alpha_i} h^{b_i}$, where α_i is randomly chosen from $\mathbb{Z}_q$. The voter also computes proof(B_{i0}). Finally, the voter determines polynomials G_i and H_i,

$$G_i(x) = \alpha_i + \alpha_{i1} x + \cdots + \alpha_{i,t-1} x^{t-1}$$
$$H_i(x) = b_i + \beta_{i1} x + \cdots + \beta_{i,t-1} x^{t-1},$$

   where the coefficients α_{il}, β_{il}, $1 \le l < t$, are chosen at random from $\mathbb{Z}_q$. Also, for these coefficients the voter computes the commitments $B_{il} = g^{\alpha_{il}} h^{\beta_{il}}$

2. The voter posts B_{i0}, proof(B_{i0}),$B_{i1}, \dots, B_{i,t-1}$ to the bulletin board.

3. All participants verify whether ballot B_{i0} is correctly formed by checking proof(B_{i0}).

4. The voter sends the respective shares $(b_{ij}, a_{ij}) = (H_i(j), G_i(j))$ to authority A_j using a private channel.

5. Each authority checks the received share (b_{ij}, a_{ij}) by verifying that

$$g^{a_{ij}} h^{b_{ij}} = \prod_{l=0}^{t-1} B_{il}^{j^l}.$$

Vote casting

To cast a vote, V_i simply posts $d_i \in \{1, -1\}$ such that $v_i = b_i d_i$ represents the desired vote.

Tallying

1. Each authority A_j posts the sum $S_j = \sum_{i=1}^{m} a_{ij} d_i$ and the sub-tally $T_j = \sum_{i=1}^{m} b_{ij} d_i$.
2. Each tallier checks the share (T_j, S_j) posted by authority A_j by verifying that

$$g^{S_j} h^{T_j} = \prod_{l=0}^{t-1} (\prod_{i=0}^{m} B_{il}^{d_i})^{j^l}$$

3. From t pairs (T_j, j) that correspond to authorities for which the shares (T_j, S_j) are correct, each tallier can compute the final tally T from the formula:

$$T = \sum_{j \in A} T_j \prod_{l \in A \setminus \{j\}} \frac{l}{l-j},$$

where A denotes a set of t correct authorities.

We assume, without loss of generality, that in a successful election the shares of every voter have been accepted by all authorities. That is, all verification in the last step of the ballot construction are successful. In case an authority receives a share that does not pass this step the authority may post the share so that anybody can verify that the share is not correct and that it corresponds to the posted encryption of step 4 of the ballot construction [CFSY96].

Theorem 4.4. Under the discrete logarithm assumption, our secret-ballot election scheme satisfies universal verifiability, computational robustness and information-theoretic privacy.

Proof. To prove universal verifiability first note that only correct ballots are counted on account of Theorem 4.1. Further, to prove that the final tally is correct, we reason as follows for each correct authority A_j. Let $G(x) = \sum_{i=1}^{m} d_i G_i(x)$ and $H(x) = \sum_{i=1}^{m} d_i H_i(x)$. By the binding property of the encryption B_{i0}[4], and the discussion of Section 4.3.3 we have:

$$\prod_{l=0}^{t-1} (\prod_{i=0}^{m} B_{il}^{d_i})^{j^l} = g^{G(j)} h^{H(j)}$$

So by the assumption that the verification in step 2 of the tallying protocol holds for (T_j, S_j), we thus have $g^{G(j)} h^{H(j)} = g^{S_j} h^{T_j}$, which implies $S_j = G(j)$ and $T_j = H(j)$ under the discrete logarithm assumption. As a consequence, the final tally T is indeed equal to $H(0)$ and thus represents the result of the election if the verification in step 2 of the tallying holds for at least t authorities. This deals with universal verifiability and robustness.

To prove privacy we note that the proof of validity is information-theoretically secure by Theorem 4.1, and that the verifiable secret sharing scheme used is information-theoretically secure by Theorem 4.3. Thus fewer than t authorities do not obtain any information about individual votes, other than what can be derived from the finally tally. □

We note that a multi-way election with L possible candidates can be performed by doing L elections in parallel, where yes-vote in election j represents a vote on candidate j. We can restrict

[4]Here we assume we use the Fiat-Shamir heuristic non-interactive Σ-protocol. If we instead use he interactive Σ-protocol we remove the need for this extra assumption.

the number of candidates a voter V_i may vote for by adding a requirement on the sum of V_i's votes. That is let v_{il} be V_i's vote in parallel election l, if we only allow a vote on at most one candidate, then we require that $\sum_{l=1}^{L} v_{il} \leq 2 - L$. Cramer et.al list numerous extensions to this approach that possibly are more efficient. We refer to [CFSY96] for the details.

5

Discussion

We will now compare the two presented voting systems with respect to security and efficiency. There are two important properties to consider when comparing the security issue of the two systems. One thing to consider is how well robustness is preserved. In other words, how secure is the voting system against attempts to cheat by voters and/or authorities running the election? The other consideration is how well privacy is preserved. In other words, how secure the encryptions are at hiding the individual votes? In the remaining we will refer to the respective voting systems by references to the papers where they first were presented. That is, we will refer to the voting system based on homomorphic encryption presented in Chapter 3 as [DGS03], while the voting system based on verifiable secret sharing presented in Chapter 4 will be refered to as [CFSY96].

We will consider robustness first. In order for a voter to cheat in [DGS03], i.e. cast a non-valid vote, he must somehow during the Σ-protocol convince the authorities that the commitment, and therefore the encryption containing the non-valid vote, is of the correct form. Let k be the length of the bit-string challenge used in the Σ-protocol. By the proof of Theorem 3.6, we know that, in order to accomplish this with probability greater then 2^{-k}, the voter must break the binding property of the commitment scheme, which is at least as hard as solving the RSA problem.

In order for the voter to cheat in [CFSY96] he must, as above, somehow during the Σ-protocol convince the authorities that the encryption containing the non-valid vote is of the correct form. If the proof is done interactively this can only happen with probability 2^{-k} by Theorem 2.1. While if the proof is done non-interactively, using the Fiat-Shamir heuristic, the cheating prover can, after answering the challenge, break the binding property to cast a non-valid vote. This is by Theorem 4.1 at least as hard as solving the discrete logarithm problem.

Comparing the two we see that, if the Σ-protocol is done interactively in both voting systems, [CFSY96] is more secure then [DGS03], since robustness in the latter relies on the RSA assumption. If both Σ-protocols are done non-interactively it is difficult to say which voting system is most secure since they rely on different intractability assumption.

Now, if some of the authorities try to cheat Theorem 4.2 ensures that robustness in [CFSY96] is preserved as long as some threshold t of the authorities are operating properly. And similary robustness in [DGS03] is preserved if at least t authorties operate properly by Theorem 3.3. This means that both voting systems are secure against corrupt authorities (or authorities hacked into) as long as at least t of the authorities are operating properly.

We will now consider privacy. In [DGS03] privacy is preserved if the cryptosystem is semantically secure, no information about the vote leaks during the performance of the Σ-protocol, and no authority can decrypt individual votes. By the proof of Theorem 3.6 it follows that privacy is preserved under the discrete logarithm assumption and as long as at most $t-1$ authorities collude or are hacked into.

In [CFSY96] privacy is preserved by the fact that the encryption with proof of validity is unconditionally hiding by Theorem 4.1. Thus privacy is preserved as long as at most $t-1$ authorities collude or are hacked into by Theorem 4.3.

Comparing the two we see that privacy in [DGS03] rely on some computational problem, while privacy in [CFSY96] is preserved in the information-theoretical sence. The extent to which lack of information-theoretic privacy is harmful may be difficult to estimate. For instance, it is hard to predict what happens if fifty-year old votes of a U.S. president are published – although breaking the encryption methods for the currently widely used security parameters will probably be much more harmful [CGS97]. We note that none of the voting systems are receipt-free since a vote-buyer or coercer can be convinced upon receiving the vote and the random number used to encrypt the vote that the voter voted as desired. However, both these voting system can be made receipt-free using the method from [HS00].

We will now compare the two voting system with respect efficiency. Since [CFSY96] is a yes/no election, while [DGS03] is a multi-way election some assumptions are needed. Let L be the number of candidates a voter may vote for. In [DGS03] the really heavy part is the Σ-protocol for proving correctness of the encrypted vote. We will assume that Σ-protocol in [DGS03] is at least as efficient as performing the proof of validity in [CFSY96] L times in parallel (which is, to say the least, a very plausible assumption). We will also assume that t authorities jointly decrypting the results in [DGS03] is about as efficient as t authorities jointly computing the final tally in [CFSY96] (which is also a very plausible assumption looking at the modular exponations needed in both cases). It then follows that [DGS03] is more efficient then [CFSY96] (in particular in a big-scale election), since the voters only have to interact with one authority in [DGS03], while each voter have to interact with each authority in [CFSY96]. We also note that Cramer et.al in [CGS97] presented a more efficient version of [CFSY96] based on homomorphic encryptions instead of verifiable secret sharing.

6

Conclusion

We have presented two fundamentally different voting systems, one based on homomorphic encryptions, and one based one verifiable secret sharing. Comparing the two we found that the one based on verifiable secret sharing was more secure with respect to both privacy and robustness. On the other hand, we found that the one based on homomorphic encryptions was the more efficient of those two.

Bibliography

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Department of Computer Science, Technion, 1996. Supervisor: Benny Chor.

[BFP$^+$01] O. Baudron, P. Fouque, D. Pointcheval, G. Poupard, and J. Stern. Practical multi-candidate election system, 2001.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 National Computer Conference*, pages 313–317. AFIPS, 1979.

[BM88] Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity class. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 544–553, New York, NY, USA, 1994. ACM Press.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli Maurer. General secure multi-party computation from any linear secret sharing scheme. Cryptology ePrint Archive, Report 2000/037, 2000.

[CDS94] Ronald Cramer, Ivan Bjerre Damgård, and Berry Schoenmakers. Proof of partial knowledge and simplified design of witness hiding protocols. *Lecture Notes in Computer Science*, 839:174–187, 1994.

[CF85] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symp. on Foundations of Comp. Science*, pages 372–382, Portland, 1985. IEEE.

[CFSY96] Ronald Cramer, Matthew Franklin, Berry Schoenmakers, and Moti Yung. Multi-authority secret-ballot elections with linear work. *Lecture Notes in Computer Science*, 1070:72–83, 1996.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Lecture Notes in Computer Science*, 1233:103+, 1997. http://citeseer.ist.psu.edu/cramer97secure.html.

[Cra96] Ronald Cramer. *Modular design of secure yet practical cryptographic protocols.* PhD thesis, University of Amsterdam, 1996.

[Cra99] Ronald Cramer. Introduction to secure computation. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 16–62, London, UK, 1999. Springer-Verlag.

[DA01] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*, pages 13–22, New York, NY, USA, 2001. ACM Press.

[Dam99] Ivan Damgård. Commitment schemes and zero-knowledge protocols. In *Lectures on Data Security, Modern Cryptology in Theory and Practice, Summer School, Aarhus, Denmark, July 1998*, pages 63–86, London, UK, 1999. Springer-Verlag.

[Dam05] Ivan Damgård. On Σ-protocols. Course material for cryptologic protocol theory at Aarhus university, 2005.

[DF01] I. Damgård and E. Fujisaki. An integer commitment scheme based on groups with hidden order, 2001. http://citeseer.ist.psu.edu/ard01integer.html.

[DGS03] Ivan Damgård, Jens Groth, and Gorm Salomonsen. The theory and implementation of an electronic voting system. In D. Gritzalis, editor, *Secure Electronic Voting*, pages 77–100. Kluwer Academic Publishers, 2003. http://citeseer.ist.psu.edu/damgaard02theory.html.

[DJ01] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Proc. of public key cryptography.* Springer Verlag LNCS series, 2001. http://citeseer.ist.psu.edu/383099.html.

[DJN03] I. Damgard, M. Jurik, and J. Nielsen. A generalization of paillier's public-key system with applications to electronic voting, 2003. http://citeseer.ist.psu.edu/damgard03generalization.html.

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

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, pages 186–194, New York, 1987. Springer-Verlag.

[FS90] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, New York, NY, USA, 1990. ACM Press.

[Gen96] R. Gennaro. *Theory and Practice of Verifiable Secret Sharing.* PhD thesis, Massachusetts Institute of Technology (MIT), 1996.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, New York, NY, USA, 1985. ACM Press.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[Gol97] Shafi Goldwasser. Multi party computations: past and present. In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6, New York, NY, USA, 1997. ACM Press.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. *Proceedings of EuroCrypt 2000*, 1807:539–556, 2000.

[Lan06] Susan Landau. Find me a hash. *Notices of the AMS*, 53(3), March 2006.

[Lip01] H. Lipmaa. Statistical zero-knowledge proofs from diophantine equations, 2001. http://citeseer.ist.psu.edu/lipmaa01statistical.html.

[LP98] Harry R. Lewis and Chistos H. Papadimitriou. *Elements of the theory of computation*. Prentice Hall, Inc., second edition, 1998.

[Nef01] C. Andrew Neff. A verifiable secret shuffle and its application to e-voting. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 116–125, New York, NY, USA, 2001. ACM Press.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–238, 1999. http://citeseer.ist.psu.edu/paillier99publickey.html.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.

[RK03] Ronald L. Rivest and Burt Kaliski. Rsa problem, December 2003.

[Saf01] Safevote. Voting system requirements, 2001. http://www.thebell.net/papers/vote-req.pdf.

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. *Lecture Notes in Computer Science*, 1666:148–164, 1999. http://citeseer.ist.psu.edu/schoenmakers99simple.html.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sti02] Douglas R. Stinson. *Cryptography, theory and practice*. Chapman and Hall/CRC, 2002.

[WFLY04] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu. Collisions for hash functions md4, md5, haval-128 and ripemd. Cryptology ePrint Archive, Report 2004/199, 2004. http://eprint.iacr.org/.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. *Lecture Notes in Computer Science*, 3621:17–36, 2005.

A

Computing discrete logarithm

We want to construct an algoritm that on input $(1+n)^i \in \mathbb{Z}^*_{n^{s+1}}$ computes i. If we define the function L by $L(b) = (b-1)/n$, then clearly we have

$$L((1+n)^i \mod n^{s+1}) = (i + \binom{i}{2}n + \cdots + \binom{i}{s}n^{s-1}) \mod n^s.$$

The idea of the algorithm is to extract the value part by part, so that we first extract $i_1 = L((1+n)^i \mod n^2) = i \mod n$. Now we can extract the rest by the following induction step: In the j'th step we know $i_{j-1} = i \mod n^{j-1}$. This means that $i_j = i_{j-1} + kn^{j-1}$ for some $0 \leq k < n$. Now consider the following equation

$$L((1+n)^i \mod n^{j+1}) = (i_{j-1} + \binom{i_j}{2}n + \cdots + \binom{i_j}{j}n^{j-1}) \mod n^j.$$

If we plugg in $i_j = i_{j-1} + kn^{j-1}$ and notice that each term $\binom{i_j}{t+1}n^t$ satisfies $\binom{i_j}{t+1}n^t = \binom{i_{j-1}}{t+1}n^t \mod n^j$ for $0 < j < t$, since the contribution from kn^{j-1} vansihes modulo n^j after multiplication by n. This means we get

$$L((1+n)^i \mod n^{j+1}) = (i_{j-1} + kn^{j-1} + \binom{i_{j-1}}{2}n + \cdots + \binom{i_{j-1}}{j}n^{j-1}) \mod n^j.$$

Now we can just rewrite the above equation to get what we want,

$$\begin{aligned} i_j &= i_{j-1} + kn^{j-1} \\ &= i_{j-1} + L((1+n)^i \mod n^{j+1}) - (i_{j-1} + \tbinom{i_{j-1}}{2}n + \cdots + \tbinom{i_{j-1}}{j}n^{j-1}) \mod n^j \\ &= L((1+n)^i \mod n^{j+1}) - (\tbinom{i_{j-1}}{2}n + \cdots + \tbinom{i_{j-1}}{j}n^{j-1}) \mod n^j. \end{aligned}$$

This equation leads to the following algorithm from [DJ01], where $a = (1+n)^i$:

```
$i := 0$;
for $j := 1$ to $s$ do
begin
  $t_1 := L(a \bmod n^{j+1})$;
  $t_2 := i$;
  for $k := 2$ to $j$ do
  begin
    $i := i - 1$
    $t_2 := t_2 i \bmod n^j$;
    $t_1 := t_1 - \frac{t_2 n^{k-1}}{k!} \bmod n^j$;
  end
  $i := t_1$;
end
```

B

Proving equality of discrete logarithms

We want to construct a Σ-protocol for proving equality of discrete logarithms. Noticing that running Schnorr's protocol for proviing knowledge of discrete logaritms two times in parallel acomplishes this we get the following protocol.

Proving equality of discrete logarithms

Common input: $u, \tilde{u}, v, \tilde{v} \in \mathbb{Z}^*_{n^{s+1}}$.
Private input for the prover: y such that $y = \log_u \tilde{u} = \log_v \tilde{v}$.

Initial message. Select r such that it shadows ey. Send $a = u^r \bmod n^{s+1}$ and $b = v^r \bmod n^{s+1}$ to the verifier.

Challenge. Select at random $e \in \{0, \ldots, 2^t - 1\}$.

Answer. Respond with $z = r + ey$,.

Verification. Accept if and only if $u^z = a\tilde{u}^e$ and $v^z = b\tilde{v}^e$.

Completness is trivial. If the prover is honest then $u^z = u^{r+ey} = u^r(u^y)^e = a\tilde{u}^e$ and similary $v^z = v^{r+ey} = v^r(v^y)^e = b\tilde{v}^e$. To show special soundness note that correct answer to two different challanges gives equations $z = r + ey$ and $z' = r + e'y$, and it follows that the prover can compute $y = (z - z')(e - e')^{-1} \bmod n^{s+1}$. Finally note that an acception conversation can be simulated by choosing e, z and random and then compute $a = u^z\tilde{u}^{-e}$ and $b = v^z\tilde{v}^{-e}$.

This Σ-protocol can be made non-interactive assuming the random oracle model. Given a cryptographically secure hash-function $\mathcal{H}$, the prover answer the challange $e = \mathcal{H}(a, b)$, resulting in z. The verifier then checks if $e = \mathcal{H}(u^z\tilde{u}^{-e}, v^z\tilde{v}^{-e})$.