NTNU
Norwegian University of
Science and Technology

Paper-based electronic voting

Anna Vederhus

Master of Science in Mathematics (for international students)
Submission date: December 2015
Supervisor: Kristian Gjøsteen, MATH

Norwegian University of Science and Technology
Department of Mathematical Sciences

Paper-based electronic voting

Anna Vederhus

December 2015

Table of Contents

Abstract

In this thesis, we present two paper-based electronic voting systems Prêt-à-Voter and Demos. We describe these in the same systematic way with new examples. Furthermore, we implement RSA cryptosystem in Prêt-à-Voter. Then, we contribute with an informal analysis of what is required both in practice and in the technical part of Demos. We present how the bulletin board must be constructed and in the analysis of the commitment scheme we propose a new way to generate the commitment key strengthening the verifiability and privacy of the system. Finally, we show how the verifiability of Demos is dependent on the practical and technical elements analysed. In specific, we show how the randomness in the voting phase contributes with entropy to the σ-protocol of ballot correctness. For a general analysis of Prêt-à-Voter and Demos this thesis should be read together with the thesis of Anna Testanière.

Sammendrag

I denne oppgaven presenterer vi to papir-baserte elektroniske stemmesystemer Prêt-à-Voter og Demos. Vi beskriver de på en lignende måte med nye eksempler og vi innfører RSA kryptosystem i Prêt-à-Voter. Videre bidrar vi med en uformell analyse av hva som kreves av den praktiske og tekniske delen av Demos. I analysen av commitment scheme foreslår vi en ny måte å genererer nøklene som styrker tilliten til systemet. Vi beskriver også hva som kreves av en bulletin board. Til slutt, viser vi hvordan verifiserbarheten til Demos er avhengig av både den praktiske og tekniske delen som analyseres. Spesielt viser vi hvordan tilfeldigheten i stemmefasen bidrar med entropi i σ-protokollen som beviser at stemmesedlene er korrekt konstruert. For en generell analyse av Prêt-à-Voter og Demos, burde denne oppgaven leses sammen med oppgaven skrevet av Anna Testanière.

Acknowledgement

I skrivende stund, har jeg endelig en opplevelse av at dette går i havn. Jeg vil takke hovedveileder Kristian Gjøsteen for hjelp i høst, det har vært fint å kunne å ha et kryptografi leksikon noen etasjer opp fra lesesalen. Jeg vil også takke Martin Strand for oppmuntrende veiledning i vår. Tusen takk til de tre jentene hjemme Torunn, Ingeborg og Kjersti som etter endt skoledag, gjør at jeg vil løpe hjem. Videre takk til mattelands innbyggere spesielt Sigurd, Marit og Anna som får meg til å ha lyst til å løpe tilbake til skolen om morgenen. Mine foreldre inspirerer meg, dere viser forbilledlig hvordan det å tilegne seg kunnskap gir mening og er gøy. Helt siden jeg kan huske, har dere engasjert dere og delt kunnskap slik at enhver av mine prosjekter har blitt så lærerike. Spesielt takk til min far som i de siste årene aldri har takket nei til en matematisk prat, uansett hvor opptatt han selv måtte være. Til slutt vil jeg takke min kjære kollega og venninne Anna, ditt intelekt, vesen og talent imponererer meg. Tusen takk for et år med latter, faglige og ikke-faglige diskusjoner, takk for at du har gjort det å skrive master til en glede!

Chapter 1

Introduction

In an election, it is crucial for the society to trust the voting system implemented. That is, the voter should be certain that when the election is done her vote is counted as intended. This property is called verifiability. Moreover, she should be able to vote as she wants without being controlled by another, and she should have the right to vote privately. This is the privacy property. Finally, the system should be accessible and understandable to every voter. This is the availability property.

These requirements, categorised into verifiability, privacy and availability are fundamental in a trustworthy system. Unfortunately, it is a quite difficult task to satisfy these properties simultaneously. Indeed, it seems that more of one gives less of the other. For example, if the voting system allows the voter to use her personal mobile-phone as a voting device, the system is definitely available, but the privacy of the vote can not be assured.

Let's now have a closer look at our traditional way of voting. In Norway, the voter must go to a polling place. There, she makes her choice in a polling booth, privately. Her paper ballot is folded so that no one can see how she voted. Then, she must register and prove to the authority that she is eligible to vote. If yes, her ballot is stamped and she puts her ballot in an urn, which will be counted at the end of the election by the authority. It is a well-thought-out system, but can we really trust this traditional way of voting? Does it fulfill the trust-requirements that we have stated?

Firstly, this system appears available since the way of voting is understandable to all and the polling places are accessible to some extent. Secondly, it seems to ensure privacy as the voter is alone in the voting booth and leaves without any receipt. But, let's not forget that nowadays the voter can use her personal electronic devices while voting and take pictures of or film her choices. This can cause vote-selling and making it possible for an adversary to coerce her.

Finally, the voter can not be certain that her ballot has been counted or even counted correctly. In other words, the voter must trust the election authority and the tellers to be honest while tallying her and all the other ballots. Miscounting and cheating occurs more

often than we think and can have great consequences. So, as we can see, this voting system has some weaknesses motivating research of a better solution. Can we make a system more secure, robust and trustworthy using new technology?

In an electronic voting system, cryptography could offer more security for instance by making it possible for the voter to verify her vote and making it more difficult for the different agents involved in a voting system to cheat. Moreover, by avoiding the "human counting", the tallying procedure could be more efficient. This could also reduce the cost of the election, especially if we consider remote electronic voting. Although not important for a trustworthy system, efficiency and cost are properties that must be taken into consideration.

However, turning to electronic voting does not only come with advantages. If we consider a remote electronic voting system, where the voter can cast her ballot from home, at work or in the bus, how can we ensure the privacy needed? How can we be certain that she was not forced or manipulated to vote in some way? This problem is so large and complicated that as per today, a remote system that ensures privacy, has not been found. We need the privacy of the voter to be kept, hence it has to be supervised, but we also want the voter to be able to verify her vote, therefore we turn to supervised electronic voting. By that, we mean having cryptographic elements making the verification possible, without losing the privacy requirement as the voting is still happening at a polling station.

Table Three voting systems with a superficial analysis of benefits and drawbacks.

	Availability	**Privacy**	**Verifiability**
Traditional Norwegian voting system	Yes. - **Accessible** to the extent that the voter needs to go to a polling station to vote. - **Understandable** to every voter.	Yes. - **Coercion resistant** because of polling booth and receipt-freeness. - **Anonymous**.	No. - The voter cannot verify that her vote was counted. - Only parts of the election process can be verified.
Remote voting system	Yes. - **Accessible** to the extent that the voter has access to Internet. - **Understandable** to every voter, may be more complicated because of the cryptographic elements and the use of the technological devices.	Difficult. - **No privacy** ensured while voting, more complicated to achieve coercion resistance. - **Anonymous** to the extent that her vote can not be traced in the system.	Yes. - The voter can verify that her vote was counted. - All parts of the election process can be verified.
Paper-based electronic voting system	Yes. - **Accessible** to the extent that the voter needs to go to a polling station to vote. - **Understandable** to every voter, may be more complicated because of the cryptographic elements.	Yes. - **Coercion resistant** because of polling booth and cryptographic receipt. - **Anonymous**.	Yes. - The voter can verify that her vote was counted. - All parts of the election process can be verified.

In chapter 2, we present the definitions and mathematical notions used in this thesis. In chapter 3, we describe the voting system Demos and explain further the setup phase, the voting phase and the tallying phase of the system. We conclude this chapter with an illustrating example. Similarly, we present in chapter 4 two versions of the voting system Prêt-à-Voter. First, we describe a version based on decryption mix-net and second, a version based on re-encryption mix-net. We also make an illustrating example of the first version.

We analyse Demos in chapter 5, 6 and 7. In chapter 5, we present and investigate the requirements needed in the voting phase first with respect to the privacy and then the verifiability. In chapter 6, we analyse two cryptographic components, namely the commitment scheme and the bulletin board. We end this thesis in chapter 7 by introducing a mathematical definition of verifiability followed by an analysis of how Demos satisfies this definition.

Chapter 2

Theory

2.1 Definitions

Numerous notions and many requirements can be used to define a voting system. We define only the notions and requirements that are used in this thesis. These definitions are informal but meant to be sufficient to read this thesis. Interested readers are welcome to read more about these notions from Clarkson, Chong and Myers [2], Juels, Catalano and Jakobsson [10], Chondros, Delis, and Gavatha et al. [7] and Kiayias, Zacharias and Zhang [11].

Electronic voting system The casting and counting of votes in a election require information and communication technologies.

Supervised voting system The voter votes at a polling station.

Remote voting system The voter does not need to go to a polling station to vote.

We concentrate on three security requirements; verifiability, availability and privacy. We present here a general definition for each of these.

Verifiability A voting system is verifiable if it fulfills one or more of the following definitions.

- **Voter Verifiability** The voter can check that her own vote is included in the tally.
- **End-to-end Verifiability** It can be checked that all votes cast are counted, that only authorized votes are counted, and that no votes are changed during counting.

Availability A voting system is available if it is both accessible and understandable.

- **Accessible** The voter is able to vote without any restriction.
- **Understandable** The voter can understand and use the system.

Privacy A voting system is private if it is coercion resistant and if it ensures anonymity.

- **Coercion Resistant** A coercer can not be certain whether the voter cooperated with him, even if they interact while she votes.
- **Anonymity** The identity of the voter and her vote can not be linked.

There is a related notion called receipt-freeness, which is a weaker version of coercion resistance.

Receipt-freeness The voter does not receive a receipt that can prove how she voted.
We remark that a system that is coercion-resistant is consequently receipt-free.

2.2 Mathematics

To ease the notation, we define $[1, n]$ to be $\{1, 2, ..., n\}$.

2.2.1 Public Key Encryption

In public key encryption [15], a message is encrypted with a public key. To decrypt this message, the corresponding private key is needed. We define public key encryption with the following notation in this thesis.

Public Key Encryption

A public cryptosystem is defined by $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ where,

- $\mathcal{P}$ denotes the set of plaintexts.
- $\mathcal{C}$ denotes the set of ciphertexts.
- $\mathcal{K}$ denotes a key generator with output (pk, sk), where pk is the public key and sk is the corresponding secret key.
- $\mathcal{E}$ denotes the set of encryption algorithms.
- $\mathcal{D}$ denotes the set of decryption algorithms.

For any (pk, sk) generated by $\mathcal{K}$ there is an $E_{pk} \in \mathcal{E}$ where E_{pk}: $\mathcal{P} \to \mathcal{C}$ and a corresponding $D_{sk} \in \mathcal{D}$, where D_{sk}: $\mathcal{C} \to \mathcal{P}$. For every $m \in \mathcal{P}$,

$$D_{sk}(E_{pk}(m)) = m$$

2.2.2 Exponential ElGamal

The exponential ElGamal is a modified version of the ElGamal cryptosystem [15]. A generator g of prime order is raised to the power of the message m, so that g^m is encrypted. While the original ElGamal is homomorphic over multiplication, this system satisfies the additive homomorphic property, $g^{m_1} \cdot g^{m_2} = g^{m_1+m_2}$. After decrypting the ciphertext, g^m is obtained. Because of the discrete logarithm problem, the message m has to be small so that it can be retrieved by known algorithms such as Shanks Algorithm or with a precomputed table. The exponential ElGamal described below can be used on a general group. To ease in reference we use the group $\mathbb{Z}_p^*$ since it is this specific version of exponential ElGamal we refer to later in this thesis.

Exponential ElGamal Public-Key cryptosystem in $\mathbb{Z}_p^*$

Let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_p^*$ with order q and generator g, where both p and q are primes. Assume computing discrete logarithms in $\mathbb{G}$ is infeasible. $\mathbb{G}$ is isomorphic to $\mathbb{Z}_q$.

- $\mathcal{P}$= $\mathbb{Z}_q$
- $\mathcal{C}$= $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$
- $(pk, sk) \leftarrow \mathcal{K}$, where $pk = (p, g, \beta)$ and $sk = e$.
  $\beta \equiv g^e \bmod p$.

Let $k, m \in \mathcal{P}$, where k is a secret random number and m is the message to encrypt. We define the encryption algorithm, E_{pk}:

$$E_{pk}(m') = (c_1, c_2),$$

where m'=g^m, $c_1 = g^k \bmod p$, $c_2 = g^m \beta^k \bmod p$.

For (c_1,c_2) $\in \mathcal{C}$, we define the decryption algorithm D_{sk}:

$$D_{sk}(c_1, c_2) = c_2({c_1}^e)^{-1} = g^m \beta^k (g^{ke})^{-1} = g^m g^{ek} (g^{ke})^{-1} = g^m = m'.$$

Then retrieving m from m' either by known algorithms or from a precomputed table.

We note that using exponential El Gamal is not problematic in a voting system since the plaintexts often are candidates from a small set, so that the discrete logarithm may be found.

2.2.3 Negligible function

One of the main goals of using a cryptographic scheme is that no adversary can break the scheme without knowing the key. However, an adversary given unbounded computational power can find the key to the scheme by brute force. Most cryptographic systems in use today assume an adversary with bounded computational power.

Negligible function

The function $neg : \mathbb{N} \to [0, 1]$ is defined as negligible if for all $c > 0$, there exists an $n > N_c$ such that,

$$neg(n) < \frac{1}{n^c}.$$

In a cryptographic system $n \in \mathbb{N}$ is the key length. We observe that, a negligible function multiplied by any polynomial $p(n)$ is still negligible, $neg(n) < \frac{1}{n^c} \cdot p(n)$. We often refer to negligible functions when defining the probability that an adversary breaks the system. If the probability of breaking the scheme is negligible, the probability stays negligible if it is repeated a polynomial number of times. For all $c > 0$, there exists an $n > N_c$ such that,

$$Pr[\text{an adversary breaks the scheme}] \leq \frac{1}{n^c} \cdot p(n).$$

2.2.4 Decisional Diffie Hellman Assumption

Decisional Diffie Hellman Assumption

Let $\mathbb{G}$ be a group generated by g. If the Decisional Diffie Hellman Assumption holds, it is infeasible to distinguish between,

$$\{(g^a, g^b, c) \mid a, b, c \in \mathbb{G}\}$$

and

$$\{(g^a, g^b, g^{ab}) \mid a, b \in \mathbb{G}\}.$$

The Decisional Diffie Hellman Assumption is stronger than assuming that computing the discrete logarithms in the group is infeasible. By knowing a or b one can compute g^{ab} and distinguish c' from c.

2.2.5 Commitment Scheme

A commitment scheme [1] is a method of committing to a value while keeping it secret to others. By this, the scheme has two properties, namely hiding and binding. The commitment gives no information about the value committed, providing the hiding property. The binding property follows because the commitment eliminate the possibility of changing the original value. To open the commitment, a secret opening value has to be revealed by the agent who preformed the commitment. Below we present a general commitment scheme.

Commitment Scheme

Let *ck* the commitment key. A commitment scheme is based on the following:

- For any $m \in \mathcal{P}$
$$Com_{ck}(m) = (c, d)$$
is the commitment/opening pair form of m, where c is the commitment value and d is the opening value.
- The opening of the commitment is presented as
$$Open_{ck}(c, d) = m \in \mathcal{P} \cup \{\perp\},$$
where $\perp$ is returned if the commitment/opening pair (c, d) does not open to any valid message in $\mathcal{P}$.

Below we show how a commitment scheme is used in practice. We assume Bob wants to commit a value m to Alice.

1. Bob generates $Com_{ck}(m) = (c, d)$, and sends c to Alice.
2. To open the commitment, Bob sends d to Alice.
3. Alice computes $Open_{ck}(c, d) = m$ and accepts the value, provided $m \neq \perp$.

In voting systems, commitment schemes can be used to ensure verifiability and privacy of a cast vote.

2.2.6 Threshold Scheme

A threshold scheme [15] is a method of sharing a secret key by dividing the information among several participants. It is called a (k, n)-threshold scheme if n is the number of participants, from which any subgroup of k participants can compute the secret key, but no group of $k-1$ participants can obtain this information.

Shamir (k, n)-Threshold Scheme

Key Distribution
Let q be a prime. A third party D wants to divide a secret key $a_0 \in \mathbb{Z}_q$ among n participants.
D chooses randomly $k-1$ elements of $\mathbb{Z}_q$, denoted $a_1, a_2, ..., a_{k-1}$ in secret.
D constructs the polynomial:

$$p(x) = a_0 + a_1 x + a_2 x + ... + a_{k-1} x^{k-1} \mod q$$

Then D chooses n non-zero elements of $\mathbb{Z}_p$, denoted x_i, for $i \in [1, n]$.
Finally, D computes

$$p(x_i) \qquad \text{for } i \in [1, n]$$

and distributes $p(x_i)$ to participant i, for $i \in [1, n]$.

Retrieving Secret Key
Only k honest participants are needed to retrieve the polynomial $p(x)$, and hence the secret key a_0. Given a set of k points, $(x_1, p(x_1)), (x_2, p(x_2)), ..., (x_k, p(x_k))$, the interpolation polynomial in the Lagrange form is the linear combination:

$$L(x) = p(x_1) l_1(x) + p(x_2) l_2(x) + ... + p(x_k) l_k(x) \mod q$$

$$\text{where,} \quad l_j(x) = \prod_{m=1, m \neq j}^{k} \frac{x - x_m}{x_j - x_m}$$

The k participants can then computes $L(0) = a_0$ in order to obtain the secret key.

The threshold scheme described above is used in voting systems to share the secret decryption key of the votes into several independent parties. This supports the verifiability requirement of a voting system, both making the whole system more robust against attacks and making it harder for malicious agents who decrypt, to manipulate votes without being caught.

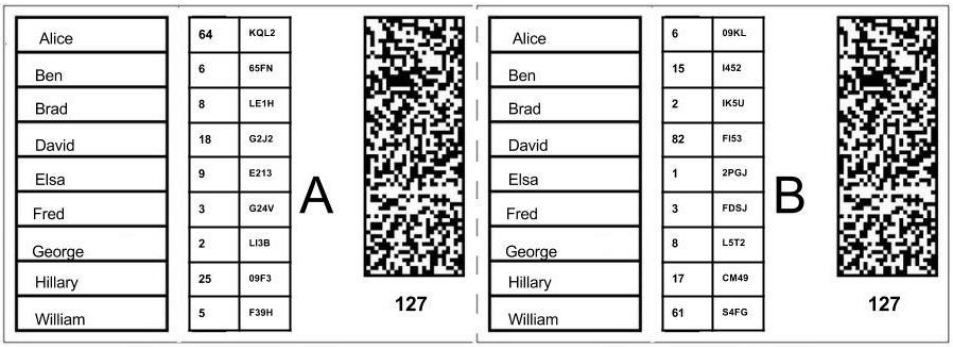
Chapter 3

Demos

3.1 Introduction

Demos [8, 11] is presented as a paper-based electronic voting system that supports verifiability and voter privacy. It has been tested in a pilot experiment during the European elections 2014 in Athens, Greece. It can be used as a supervised or a remote voting system. We present the supervised version.

At the polling station, the voter receives one ballot including left-hand side A and right-hand side B, both containing the candidate list in alphabetical order. For each candidate there is a corresponding vote-code and a code-receipt. They are both randomly chosen for each side, and they are unique for each candidate. The voter must separate side A and side B and choose randomly between them: one side is used for voting and the other one for verification.



A ballot including left-hand side A and right-hand side B, having ballot number 127.

Without loss of generality, we assume that she chooses side A for voting and side B for verification. In order to cast her vote, she first scans the QR-code on side A in the voting machine. The names of the candidates appear on the screen. She selects the candidate of

her choice and the corresponding code-receipt will appear. She can immediately verify that the code-receipt on the screen corresponds to the code-receipt next to her candidate on the paper-ballot. She writes down the vote-code corresponding to the candidate of her choice from side A and keeps side B for verification. Side A must be destroyed, so no one can figure out which candidate corresponds to her vote-code.

In advance, the election authority in charge of the election has committed to all the information on side A and on side B, so that these can not be changed during the election. The voter can verify that her vote was counted correctly by logging in with her ballot number on the election website. There, all the vote-codes from side A appear in a random order, and one of the vote-codes is marked. To provide coercion-resistance, the names of the candidates do not appear on the screen. Because of that, she can not be sure that this marked vote-code corresponds to the correct candidate.

To verify the system, the voter does two things. First, she checks that the vote-code marked on the screen is the same as she has written down from side A. Moreover, because the voter choses side A to vote with, the commitment on side B will be opened and available on the website. She verifies that the side B on the screen is exactly the same as the side B she has kept.

The key idea of the system is that the vote-codes of the candidates are different from ballot to ballot. This makes it impossible for a third party to guess which candidate corresponds to the vote-code written on the website. This contributes to the privacy of the system. Furthermore, each voter verifies the correctness of a random side of their ballot on the election website. If the information on the side appearing on the screen is correct, so should the information on the other side. This gives assurance that the ballots are constructed in a correct way contributing to the verifiability of the system.

3.2 Cryptographic description

We present in this section the cryptographic description of Demos. The voting system is based on commitments made before the election of both the vote-codes and the candidates. Most of the work of the election authority is done and published in advance in a committed form, so that they minimize the work after the voting phase. This gives assurance to voters and candidates that the election authority does not cheat during the process.

The code-receipt is not included in the article presenting the mathematical description of the system. When the voter cast her ballot by submitting her vote-code, the corresponding code-receipt appears on the screen. This assures the voter she votes correctly, since the code-receipts are unique. The code-receipt does not seem to have other purposes and is therefore omitted in this thesis.

The bulletin board often takes form of an election website in a voting system. From now on, we use the more technical notion bulletin board instead of election website.

Additionally, the system is presented for an election where the voter votes for one candidate only. The system may also be implemented for an election where the voter can vote for multiple candidates.

3.2.1 Setup phase

There are three different types of agents involved in the voting system: the election authority, the bulletin board and the voters. Demos stresses that the election authority can be split into different parties.

- The election authority generates the setup phase, the ballots and their commitments, tallies the votes and publishes proofs of his honest work.
- The bulletin board passively provides storage of information used for verification and results.
- The voter votes for the candidate of her choice, and can verify that her vote was counted correctly.

First, the election authority includes the set of voters, $\mathcal{V} = \{V_1, V_2, ..., V_l\}$, and the set of candidates, $\mathcal{T} = \{T_1, T_2, ..., T_m\}$ in the voting system. Then, the election authority generates the commitment key, ck, for the commitment scheme.

Commitment Key Generation

The commitment key generation in Demos is based on a group over an elliptic curve. The elliptic curve domain parameters are pk:=(p, a, b, g, q), where

- p is a prime specifying the field $\mathbb{F}_p$.
- $a, b \in \mathbb{F}_p$ define the elliptic curve $E(\mathbb{F}_p)$ by the equation, $E : y^2 = x^3 + ax + b$ mod p.
- $g = (x_0, y_0)$ is an element in $E(\mathbb{F}_p)$ with prime order q.
- $\mathbb{G}$ is the group generated by g and is isomorphic to $\mathbb{Z}_q$.
- it is assumed the Decisional Diffie Hellman Assumption holds over $\mathbb{G}$.

Let ω be a random element in $\mathbb{Z}_q$ and $h = g^\omega$.
The commitment key is $ck = (pk, h)$.

Encoded candidate The election authority encodes the candidates in a number system to facilitate the tallying. In the number system with base $l + 1$, where l is the number of voters and m is the number of candidates, the encoding of candidates is denoted by

$$T_i \leftarrow (l + 1)^{i-1}, \quad i \in [1, m].$$

Thus, the pairs $(T_1, (l + 1)^0), (T_2, (l + 1)^1), ..., (T_m, (l + 1)^{m-1})$ are obtained. Note that the encoding of candidates is the same for all the ballots.

Generating a ballot We now describe the process of generating a ballot. A ballot must include a unique ballot number, two sides A and B containing the list of the candidates in alphabetical order $(T_1, T_2, ..., T_m)$ with their corresponding vote-codes $(C_1, C_2, ..., C_m)$ and QR-codes containing this information.

Generating a ballot

A ballot consists of side A, s^A, and side B, s^B. To create a ballot the election authority does the following:

1. Selects a unique ballot number, denoted *BN*.
2. Selects random vote-codes, $C_i^A \in \mathbb{Z}_q$ for $i \in [1, m]$, unique for each candidate, $T_i \in \mathcal{T}$, on side A. Similarly, selects random $C_i^B \in \mathbb{Z}_q$ on side B.
3. Generates the ballot, $s = (BN, s^A, s^B)$, where $s^A = \{(T_i, C_i^A)\}_{i \in [1,m]}$ and $s^B = \{(T_i, C_i^B)\}_{i \in [1,m]}$.

Commitment scheme To ensure privacy and verifiability, the encoded candidates together with the vote-codes must be kept secret and unchanged. Demos uses therefore a commitment scheme during the setup phase which commits the election authority to the encoding and the vote-codes while keeping these secret to others during the election. The commitments are opened during the tallying phase. As wanted, the commitment scheme gives an assurance that both the encoding of the candidates and the vote-codes remain secret and unchanged. The election authority implements a commitment scheme based on the discrete logarithm problem using the commitment key ck.

Commitment Scheme based on exponential ElGamal

Let $ck = (pk, h)$ be the commitment key defined above. For $m, t \in \mathbb{Z}_q$ where t is random:

$$Com_{ck}(m) = (c, d) = ((g^t, g^m \cdot h^t), t)$$

is the commitment/opening pair form of m.
The commitment/opening pair is homomorphic under operation:

$$Com_{ck}(m_1) \cdot Com_{ck}(m_2) = Com_{ck}(m_1 + m_2)$$

For $m_1, m_2 \in \mathcal{P}$.

Vote-code commitment The election authority generates random $t_i^A, t_i^B \in \mathbb{Z}_q$ and computes the vote-code commitment/opening pairs for $i \in [1, m]$:

$$Com_{ck}(C_i^A) = (c(C_i^A), d(C_i^A)) = ((g^{t_i^A}, g^{C_i^A} \cdot h^{t_i^A}), t_i^A)$$

$$Com_{ck}(C_i^B) = (c(C_i^B), d(C_i^B)) = ((g^{t_i^B}, g^{C_i^B} \cdot h^{t_i^B}), t_i^B)$$

Encoded candidate commitment The election authority generates random $r_i^A, r_i^B \in \mathbb{Z}_q$ and computes the encoded candidate commitment/opening pairs for $i \in [1, m]$:

$$Com_{ck}((l+1)^{i-1}) = (c((l+1)^{i-1}), d((l+1)^{i-1})) = ((g^{r_i^A}, g^{(l+1)^{i-1}} \cdot h^{r_i^A}), r_i^A)$$

$$Com_{ck}((l+1)^{i-1}) = (c((l+1)^{i-1}), d((l+1)^{i-1})) = ((g^{r_i^B}, g^{(l+1)^{i-1}} \cdot h^{r_i^B}), r_i^B)$$

Random permutation of the order of the candidates To support privacy of the vote on the bulletin board, the election authority selects random permutations which shuffle the order of the vote-codes on side A and side B for each ballot. Indeed, if the vote-codes are not shuffled, the position of the marked vote-code on the bulletin board reveals the vote. Without information of the permutation, the privacy of the voter is kept during this phase. For each ballot, the random permutations on side A and side B are:

$$\pi^A : [1, m] \to [1, m]$$

$$\pi^B : [1, m] \to [1, m]$$

The new position of the ith vote-code is denoted by $\pi^A(i)$ and $\pi^B(i)$, respectively.

Finally, the election authority publishes all the information needed on the bulletin board. The opening $(d(C_{\pi(i)}), d((l+1)^{i-1}))$ for $\pi(i) \in [1, m]$ is kept secret.

Published on the bulletin board

- The set of candidates $\mathcal{T}$.
- The set of voters $\mathcal{V}$.
- The commitment key, ck.
- For side A and side B of each ballot; the ballot number, *BN*, and the pair of vote-code/encoded candidate in committed form:

  $$(c(C_{\pi(i)}), c((l+1)^{i-1})) \quad \text{for } \pi(i) \in [1, m]$$

  in the permutation order, π^A, π^B, respectively.

3.2.2 Voting phase

During the voting phase, the voter chooses randomly between side A and side B by tossing a coin. Without loss of generality, let's assume she votes with side A. She chooses the candidate $T_i \in \mathcal{T}$ of her choice, by selecting the corresponding vote-code C_i^A. The vote cast is $V_{\text{cast}} = (BN, s^A, C_i^A)$ and the receipt obtained is $V_{\text{receipt}} = (BN, s^B, C_i^A)$.

3.2.3 Tallying phase

After the vote is cast, it must be recorded in the system. The other side can be published together with its secret information for verification on the bulletin board.

Retrieving information from a vote

Without loss of generality, we assume the vote cast is $V_{\text{cast}} = (BN, s^A, C_i^A)$.
Then the election authority follows the procedure:

1. Publishes (V_{cast}, s^B) to the bulletin board and opens the commitments of side B.
2. Matches the vote-code C_i^A with the permuted vote-code: $C_{\pi(i)}^A \leftarrow C_i^A$.
3. Marks $C_{\pi(i)}^A$ as voted and sends the corresponding commitment $c((l+1)^{i-1})$ to tallying.

Now, the election authority has sent all the votes in their encoded commitment form to the tally. Due to their homomorphic property, it is the commitments of the encoded candidates that are counted under the tallying.

Tallying of the ballots

We denote *C* the product of all the encoded candidates commitments for all the votes cast l. That is,

$$C = \prod_{j=1}^{l} c((l+1)^{i_j-1}) = c((l+1)^{i_1-1} + (l+1)^{i_2-1} + ... + (l+1)^{i_l-1}),$$

where i_j corresponds to the candidate choice of voter V_j. The election authority publishes C on the bulletin board with its corresponding opening. We set $Open_{ck}(C) = T$. The election result, R, is calculated in the number system with base $l+1$ by the following algorithm. For $i \in [1, m]$:

- $x_i = T \bmod (l+1)$
- $T = (T - x_i)/(l+1)$
- Return $R = (x_1, x_2, ..., x_m)$

After tallying the ballots, the election authority publishes the result $R = (x_1, x_2, ..., x_m)$, where x_i is the number of votes for the candidate T_i.

3.3 Example

We make a simplified example of the system for a better understanding.

Setup phase The election authority includes two voters, V_1 and V_2 and three candidates namely Anna, Marit and Sigurd in the voting system. Additionally, the election authority encodes the candidates. Anna is encoded to $(2+1)^0 = 1$, Marit is encoded to $(2+1)^1 = 3$ and Sigurd is encoded to $(2+1)^2 = 9$.

Generation of ballots

The election authority proceed with the following:

Generation of ballot s_1

1. Selects *BN*$= 23$.
2. Respectively, for Anna, Marit and Sigurd,
   - Selects random vote-codes $C_1^A = 241$, $C_2^A = 756$ and $C_3^A = 345$ for side A.
   - Selects random vote-codes $C_1^B = 123$, $C_2^B = 385$ and $C_3^B = 946$ for side B.
3. Generates the ballot $s_1 = (23, s^A, s^B)$, where
   - $s^A = ((\text{Anna}, 241), (\text{Marit}, 756), (\text{Sigurd}, 345))$
   - $s^B = ((\text{Anna}, 123), (\text{Marit}, 385), (\text{Sigurd}, 946))$

Generation of ballot s_2

1. Selects *BN*$= 84$.
2. Respectively, for Anna, Marit and Sigurd,
   - Selects vote-codes $C_1^A = 256$, $C_2^A = 486$ and $C_3^A = 542$ for side A.
   - Selects vote-codes $C_1^B = 383$, $C_2^B = 430$ and $C_3^B = 639$ for side B.
3. Generates the ballot $s_2 = (84, s^A, s^B)$, where
   - $s^A = ((\text{Anna}, 256), (\text{Marit}, 486), (\text{Sigurd}, 542))$
   - $s^B = ((\text{Anna}, 383), (\text{Marit}, 430), (\text{Sigurd}, 639))$

Then, the election authority selects random permutations to shuffle the vote-codes on each ballot,

Ballot number 23: $\pi_1^A : (1,2,3) \to (3,2,1)$ and $\pi_1^B : (1,2,3) \to (1,2,3)$

Ballot number 84: $\pi_2^A : (1,2,3) \to (2,1,3)$ and $\pi_2^B : (1,2,3) \to (3,2,1)$

The vote-code and encoding for each side of each ballot are paired up. Finally, the election authority generates the commitment/opening pair of the vote-codes and the encodings of the candidates. The commitments in their permutation order are posted on the bulletin board while the corresponding openings are kept secret by the election authority.

Published on the bulletin board

- The set of candidates: Anna, Marit, Sigurd.
- The set of voters: V_1 and V_2.
- The commitment key, ck.

Ballot number 23:		**Ballot number 84**:	
Side A:	Side B:	Side A:	Side B:
$(c(345), c(9))$	$(c(123), c(1))$	$(c(486), c(3))$	$(c(639), c(9))$
$(c(756), c(3))$	$(c(385), c(3))$	$(c(256), c(1))$	$(c(430), c(3))$
$(c(241), c(1))$	$(c(946), c(9))$	$(c(542), c(9))$	$(c(383), c(1))$

Voting phase After these computations by the election authority, the voting phase can start.

- V_1 picks the ballot $s_1 = (23, s^A, s^B)$, flips a coin, and votes with side A, s^A, for Anna.
  1. The vote cast is $V_{\text{cast}} = (23, s^A, 241)$.
  2. The vote receipt received is $V_{\text{receipt}} = (23, s^B, 241)$.
- V_2 picks the ballot $s_2 = (84, s^A, s^B)$, flips a coin, and votes with side B, s^B, for Sigurd.
  1. The vote cast is $V_{\text{cast}} = (38, s^B, 639)$.
  2. The vote receipt received is $V_{\text{receipt}} = (38, s^A, 639)$.

Tallying phase After the voting is done, the election authority retrieves information of the vote.

Retrieving information from the votes

The election authority:

1. Publishes $V_{\text{cast}} = (23, s^A, 241)$ and $V_{\text{cast}} = (38, s^B, 639)$ on the bulletin board.
2. Opens the commitments of side B for ballot 23 and side A for ballot 38 and the commitments of the vote-code cast, 241 and 639.
3. Matches the vote-codes with their corresponding encoded candidate commitments $(241, c(1))$ and $(639, c(9))$ and mark them as voted.
4. Sends the encoded candidate commitments $c(1)$ and $c(9)$ to tallying.

Now the election authority tallies the votes.

Tallying of votes

The tallying is done homomorphically by computing the product of the commitment/opening pair,

$$Com_{ck}(1) \cdot Com_{ck}(9) = Com_{ck}(1+9) = Com_{ck}(10) = (c(10), d(10))$$

The election authority publishes c(10) along with its opening, d(10), on the bulletin board, hence $T = Open_{ck}(c(10), d(10)) = 10$ is now known.

The election result, R is calculated in the number system with base $l + 1 = 3$ by the algorithm presented earlier:

- $x_1 = T \bmod l + 1 = 10 \bmod 3 = 1$
- $T = (T - x_1)/(l + 1) = 9/3 = 3$
- $x_2 = T \bmod l + 1 = 3 \bmod 3 = 0$
- $T = (T - x_2)/(l + 1) = 3/3 = 1$
- $x_3 = T \bmod l + 1 = 1 \bmod 3 = 1$

$\Rightarrow R = (x_1, x_2, x_3) = (1, 0, 1)$.

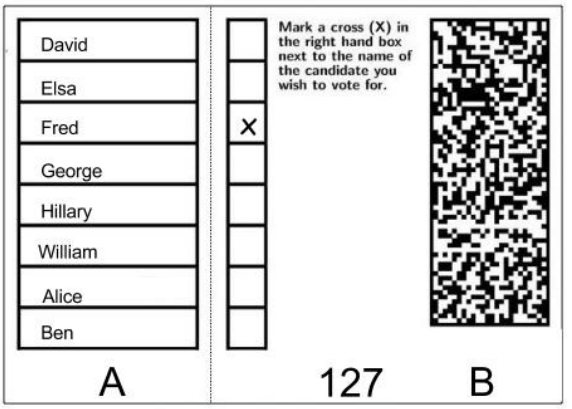
The result states that Anna got one vote, Marit got zero votes and Sigurd got one vote.

Chapter 4

Prêt-à-voter

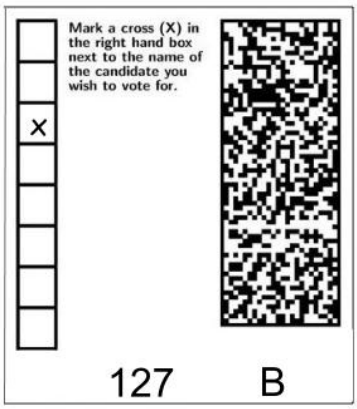
4.1 Introduction

Prêt-à-voter is a supervised electronic voting system based on paper ballots. At the polling station, the voter receives a ballot with two sides, for simplicity called side A and side B. Side A contains the list of the candidates, written in an alphabetical order with a cyclic shift, for each ballot. The voter marks a cross next to the candidate of her choice on side B. Then, she has to divide the two parts.



A completed ballot form with side A and side B having ballot number 127.

Side A must be destroyed at once, so no one knows the order of the candidate list on her ballot. Side B, however, is signed and scanned to be counted, then it is kept by the voter as a receipt.



Side B of the ballot form, scanned to be counted and kept as receipt.

Side B contains a cross marking a position but not a candidate. She can later verify that her vote was counted correctly by entering her ballot number on the election website. If nothing went wrong, her exact side B should appear on the website. The key idea of the system is that the candidate list is random, varying from ballot to ballot. This makes it impossible for a third party to guess which exact candidate order corresponds to the receipt, side B.

Prêt-à-voter is a family of voting systems [12, 14, 6, 3, 13]. We concentrate in this thesis on the article summarising Prêt-à-voter from 2010 [13], presenting two different ways of designing the ballot forms, and how these can be tallied. First, we present the design based on decryption mix-net used when the ballots are pre-printed. Then, we present re-encryption mix-net, used when the ballots are printed on-demand.

4.2 Cryptographic description by decryption mix-net

The ballot is constructed in a way so that the encrypted vote, side B with the marked cross, can be decrypted. Simply explained, the QR-code contains encrypted information of which box corresponds to which candidate. So that when the vote is cast, it is possible to retrieve which candidate the marked cross corresponds to.

4.2.1 Setup phase

There are four different types of agents involved in the decryption mix-net version of Prêt-à-voter: the election authority, the mix-servers, the bulletin board and the voters.

- The election authority includes the set of voters, the set of candidates and the set of mix-servers in the voting system and designs the ballots by using public keys generated by the mix-servers.
- The mix-servers generate key-pairs both for encryption of ballots and decryption of votes and shuffle all the ballots.

- The bulletin board passively provides storage of information used for verification and results.
- The voter votes for the candidate of her choice, and can verify that her vote was counted correctly.

First, the election authority includes the set of voters, $\mathcal{V} = \{V_1, V_2, ..., V_l\}$, the set of candidates, $\mathcal{T} = \{T_1, T_2, ..., T_m\}$ and the set of mix-servers $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ in the voting system. Each mix-server $S_i \in \mathcal{S}$ generates two pairs of keys, $(pk_{i,1}, sk_{i,1})$ and $(pk_{i,2}, sk_{i,2})$ using RSA key generation [15].

Key Generation using RSA

A key pair, (pk, sk), is generated using ***RSA***. The mix-server proceeds with the following:

1. Generates two large primes, p and q, such that $p \neq q$ and differ in length by a few digits.
2. Calculates $n = pq$ and $\phi(n) = (p-1)(q-1)$, where ϕ is the Eulers' totient function.
3. Chooses a random $b \in [1, \phi(n)]$ such that $gcd(b, \phi(n)) = 1$.
4. Calculates $a = b^{-1} \bmod \phi(n)$.

The key pair (pk, sk) is now generated where, $pk = (n, b)$ and $sk = (p, q, a)$.

To prepare against failing mix-servers, the keys are shared among all the mix-servers in a threshold scheme presented in section 2.2.6.

Encryption We denote the alphabetical ordered candidate list $(T_1, T_2, ..., T_m)$. First the election authority chooses a random shift $\alpha_{0,2}$ of this list, denoted $(T_1, T_2, ..., T_m)_{\alpha_{0,2} \bmod m}$ which is the order used during the tallying of the votes.

Assuming there are k mix-servers, the election authority continues by selecting $2k$ random seed values $\{t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}, ..., t_{k,1}, t_{k,2}\}$. With a hash of the seed $t_{1,1}$, denoted $h_{1,1}$, the election authority shifts the candidate list $(T_1, T_2, ..., T_m)_{\alpha_{0,2} \bmod m}$ to obtain $(T_1, T_2, ..., T_m)_{\alpha_{0,2}+h_{1,1} \bmod m}$. He repeats this procedure, and the candidate list printed on the ballot is $(T_1, T_2, ..., T_m)_{\alpha_{0,2}+\Sigma_{i=1}^{k}(h_{i,1}+h_{i,2}) \bmod m} = (T_1, T_2, ..., T_m)_{\alpha_{0,2}+h \bmod m}$.

To not reveal how the candidate list has been shifted, the election authority encrypts $\alpha_{0,2}$ and the random seed values $\{t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}, ..., t_{k,1}, t_{k,2}\}$ layers by layers into an onion using the $2k$ public keys generated by the mix-servers. How this is done, is explained thoroughly later. In this design any public cryptosystem can be implemented, but we present a case using the *RSA public cryptosystem*.

RSA public cryptosystem

Let $n = pq$ where, p and q are primes.

1. $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$.
2. $\mathcal{K} = (pk, sk) = ((n, b), (p, q, a))$ where, $ab \equiv 1 \bmod \phi(n)$.
3. The encryption algorithm $E_{pk} \in \mathcal{E}$ is,
$$E_{pk}(m) = m^b \bmod n.$$
4. The decryption algorithm $D_{sk} \in \mathcal{D}$ is,
$$D_{sk}(c) = c^a \bmod n.$$

For each ballot the election authority does the following:

1. Selects randomly $\alpha_{0,2} \in \mathbb{Z}$, which modulo m represents the cyclic shift of a candidate list, $(T_1, T_2, ..., T_m)_{\alpha_{0,2} \bmod m}$.
2. Selects randomly $t_{i,1} \in \mathbb{Z}_{n_{i,1}}$ and $t_{i,2} \in \mathbb{Z}_{n_{i,2}}$ for $i \in [1, k]$:
   (a) Shifts the candidate list by a hash of $t_{i,1}$, $h_{i,1} = H(t_{i,1}) \bmod m$ and $t_{i,2}$, $h_{i,2} = H(t_{i,2}) \bmod m$ such that,
   $$(T_1, T_2, ..., T_m)_{\alpha_{0,2} + \Sigma_{i=1}^{k}(h_{i,1} + h_{i,2}) \bmod m}.$$
   (b) Encrypts the seeds $t_{i,1}$ of the shift $h_{i,1}$ and $t_{i,2}$ of the shift $h_{i,2}$ with the public key $pk_{i,1} = (n_{i,1}, b_{i,1})$ and $pk_{i,2} = (n_{i,2}, b_{i,2})$ respectively such that,
   $$\alpha_{i,1} = E_{pk_{i,1}}(t_{i,1}, \alpha_{i-1,2}) = (t_{i,1}, \alpha_{i-1,2})^{b_{i,1}} \bmod n_{i,1} = (t_{i,1}^{b_{i,1}}, \alpha_{i-1,2}^{b_{i,1}}) \bmod n_{i,1},$$
   $$\alpha_{i,2} = E_{pk_{i,2}}(t_{i,2}, \alpha_{i,1}) = (t_{i,2}, \alpha_{i,1})^{b_{i,2}} \bmod n_{i,2} = (t_{i,2}^{b_{i,2}}, \alpha_{i,1}^{b_{i,2}}) \bmod n_{i,2}.$$

From the initial list $(T_1, T_2, ..., T_m)_{\alpha_{0,2}}$, the candidate list is cyclically shifted by,

$$h = \Sigma_{i=1}^{k} h_{i,1} + h_{i,2} \bmod m.$$

The onion of the initial order $\alpha_{0,2}$ and the seed value $t_{i,j}$ of the shift $h_{i,j}$ for $i \in [1, k]$ and for $j \in [1, 2]$ is,

$$\alpha = \alpha_{k,2} = E_{pk_{k,2}}(t_{k,2}, E_{pk_{k,1}}(..., E_{pk_{1,2}}(t_{1,2}, E_{pk_{1,1}}(t_{1,1}, \alpha_{0,2})))).$$

Generating a ballot We now describe the process of generating a ballot. A ballot must include two sides, A and B. Side A contains the list of the candidates in an alphabetical order $(T_1, T_2, ..., T_m)$ with a cyclic shift $\alpha_{0,2} + h \bmod m$. Side B contains the ballot number, boxes aligned with the candidate list and the QR-code containing encrypted information.

Generating a ballot

A ballot consists of side A, s^A, and side B, s^B. To create a ballot the election authority does the following:

1. Selects a unique ballot number, denoted *BN*.
2. Selects an initial order $\alpha_{0,2}$ and computes the cyclic shift h of the candidate list $(T_1, T_2, ..., T_m)_{\alpha_{0,2} \bmod m}$, obtaining $(T_1, T_2, ..., T_m)_{\alpha_{0,2}+h \bmod m}$.
3. Encrypts the initial order of the candidate list $\alpha_{0,2}$ together with the seeds $t_{i,j}$ of the shifts $h_{i,j}$ for $i \in [1, k]$ and $j \in [1, 2]$, into the onion α which is encoded into a QR-code.
4. Generates the ballot, $s = (s^A, s^B)$ where, $s^A = (T_1, T_2, ..., T_m)_{\alpha_{0,2}+h \bmod m}$ and $s^B = \{$*BN*,QR$\}$.

4.2.2 Voting phase

During the voting phase, the voter receives a ballot $s = (s^A, s^B)$. She chooses the candidate of her choice, $T_i \in \mathcal{T}$, and marks a cross in the corresponding box. The vote cast is $V_{\text{cast}} = (s^B, \beta)$ where β is the position of the cross. She destroys s^A and receives the receipt, $V_{\text{receipt}} = (s^B, \beta)$. Note that $V_{\text{cast}} = V_{\text{receipt}}$ in Prêt-à-voter.

4.2.3 Tallying phase

Decryption The encrypted vote, V_{cast}, contains the position of the cross and a corresponding onion where the initial order of the candidate list is encrypted (α, β). To decrypt their part, mix-servers use their two secret keys to peel off two layers of the onion. Instead of shifting the order of the list as the election authority did during the encryption, the mix-server shifts the place of the cross. He then passes the partly-decrypted onion and the new cross to the next mix-server who follows the same procedure. The result is a specific position, $\beta_{0,2}$, of the cross, together with a specific order of the candidate list $\alpha_{0,2}$. These are matched to obtain the vote.

We define $\beta = \beta_{k,2}$ to be the position of the cross made by the voter on the ballot. For every ballot, each mix-server $S_i \in \mathcal{S}$ does the following:

1. Retrieves the pair $(\alpha_{i,2}, \beta_{i,2})$.
2. Decrypts $\alpha_{i,2}$ one layer, using his secret key $sk_{i,2}$, $D_{sk_{i,2}}(\alpha_{i,2}) = (t_{i,2}, \alpha_{i,1})$.
3. Computes the hash of $t_{i,2}$, $h_{i,2} = H(t_{i,2}) \bmod m$.
4. Calculates the cyclic shift of the cross, $\beta_{i,1} = \beta_{i,2} - h_{i,2}$.
5. Obtains the pair $(\alpha_{i,1}, \beta_{i,1})$.
6. Decrypts $\alpha_{i,1}$ one layer, using his secret key $sk_{i,1}$, $D_{sk_{i,1}}(\alpha_{i,1}) = (t_{i,1}, \alpha_{i-1,2})$.

7. Computes the hash of $t_{i,1}$, $h_{i,1} = H(t_{i,1}) \bmod m$.

8. Calculates the cyclic shift of the cross, $\beta_{i-1,2} = \beta_{i,1} - h_{i,1}$.

9. Obtains the pair $(\alpha_{i-1,2}, \beta_{i-1,2})$.

After $2k$ decryption, the pair $(\alpha_{0,2}, \beta_{0,2})$ is obtained. The initial order of the candidate list, $(T_1, T_2, ..., T_m)_{\alpha_{0,2} \bmod m}$, can be calculated and matched with the cross in position $\beta_{0,2}$.

Mix-net Above we have described how a mix-server decrypts his part of the onion for one vote. To provide privacy of the votes in the tallying phase, each mix-server must also shuffle the collection of pairs $(\alpha_{i,1}, \beta_{i,1})$ and $(\alpha_{i-1,2}, \beta_{i-1,2})$. Each mix-server $S_i \in \mathcal{S}$ retrieves a collection $L_{i,2} = (B^1_{i,2}, B^2_{i,2}, ..., B^l_{i,2})$ from the bulletin board where, $B^j_{i,2} = (\alpha^j_{i,2}, \beta^j_{i,2})$ is the vote from V_j, for $j \in [1, l]$. He decrypts, and publishes the decrypted pairs in permuted order on the bulletin board. This is done twice for each mix-server, using their two secret keys. The next mix-server continues with the same procedure, decrypting each ballot with his secret keys and shuffling the decrypted collection of pairs.

The mix-server S_i does the following using his secret keys $sk_{i,2}$ and $sk_{i,1}$:

1. Retrieves the collection $L_{i,2} = (B^1_{i,2}, B^2_{i,2}, ..., B^l_{i,2})$ from the bulletin board.

2. Decrypts $D_{sk_{i,2}}(B_{i,2})^j = B^j_{i,1}$ for $j \in [1, l]$.

3. Selects a permutation, π, randomly.

4. Permutes the collection such that, $L_{i,1} = (B^{\pi(1)}_{i,1}, B^{\pi(2)}_{i,1}, ..., B^{\pi(l)}_{i,1})$.

5. Decrypts $D_{sk_{i,1}}(B_{i,1})^j = B^j_{i-1,2}$ for $j \in [1, l]$.

6. Selects a permutation, π, randomly.

7. Permutes the collection such that, $L_{i-1,2} = (B^{\pi(1)}_{i-1,2}, B^{\pi(2)}_{i-1,2}, ..., B^{\pi(l)}_{i-1,2})$.

8. Publishes $L_{i-1,2}$ to the bulletin board.

4.3 Example

We make a simplified example of the system for a better understanding.

Setup phase First, the election authority includes one voter, V, three candidates $\mathcal{T} = \{\text{Anna, Marit, Sigurd}\}$ and two mix-servers, $\mathcal{S} = \{S_1, S_2\}$ in the voting system. Since there are two mix-servers, the election authority generates the ballot with four random values $\{t_{1,1}, t_{1,2}, t_{2,1}, t_{2,2}\}$. The mix-servers, S_1 and S_2, generate two public and two secret keys, denoted $(pk_{1,1}, sk_{1,1}), (pk_{1,2}, sk_{1,2})$ and $(pk_{2,1}, sk_{2,1}), (pk_{2,2}, sk_{2,2})$ respectively.

Encryption The election authority encrypts the ballot of the voter in the following way:

1. Selects randomly $\alpha_{0,2} \in \mathbb{Z} = 2$, which represents the shift of the alphabetical ordered candidate list,

$$\text{(Anna, Marit, Sigurd)}_2 = \text{(Marit, Sigurd, Anna)}.$$

2. Selects randomly $t_{i,j} \in \mathbb{Z}_{n_{i,j}}$ for $i \in [1,2]$ and $j \in [1,2]$,

   (a) Computes

$$h_{1,1} = H(t_{1,1}) = 2 \bmod 3$$
$$h_{1,2} = H(t_{1,2}) = 2 \bmod 3$$
$$h_{2,1} = H(t_{2,1}) = 1 \bmod 3$$
$$h_{2,2} = H(t_{2,2}) = 0 \bmod 3$$

   (b) Encrypts the seed $t_{i,j}$ of the shift $h_{i,j}$ with the public key $pk_{i,j} = (n_{i,j}, b_{i,j})$ for $i \in [1,2]$ and $j \in [1,2]$ such that,

$$\alpha_{1,1} = E_{pk_{1,1}}(t_{1,1}, \alpha_{0,2})$$
$$\alpha_{1,2} = E_{pk_{1,2}}(t_{1,2}, \alpha_{1,1})$$
$$\alpha_{2,1} = E_{pk_{2,1}}(t_{2,1}, \alpha_{1,2})$$
$$\alpha_{2,2} = E_{pk_{2,2}}(t_{2,2}, \alpha_{2,1})$$

3. The initial list (Marit, Sigurd, Anna) is cyclically shifted by, $h = \Sigma_{i=0}^{3}(h_i) = 5 = 2$ mod 3,

$$\text{(Marit, Sigurd, Anna)}_2 = \text{(Sigurd, Anna, Marit)}.$$

   The onion of the initial order $\alpha_{0,2}$ and the seed value $t_{i,j}$ of the shift $h_{i,j}$ for $i \in [1,2]$ and $j \in [1,2]$ is,

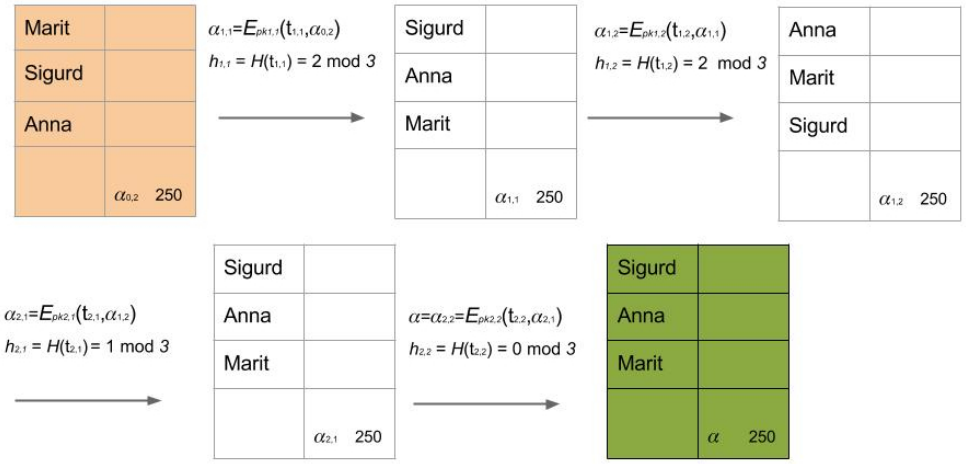
$$\alpha = \alpha_{2,2} = E_{pk_{2,2}}(t_{2,2}, E_{pk_{2,1}}(t_{2,1}(E_{pk_{1,2}}(t_{1,2}, E_{pk_{1,1}}(t_{1,1}, \alpha_{0,2}))))).$$

Generating a ballot Now the election authority generates the ballot with the encrypted information above.

Generating a ballot

A ballot consists of side A, s^A, and side B, s^B. To create the ballot the election authority does the following:

1. Selects a unique ballot number, denoted *BN*$= 250$.
2. Selects the initial order $\alpha_{0,2} = 2$ and computes the cyclic shift $h = 2$ of the candidate list (Marit, Sigurd, Anna)$_2$, obtaining (Sigurd, Anna, Marit).
3. Encrypts the initial order of the candidate list $\alpha_{0,2} = 2$ together with the seeds $t_{i,j}$ of the shifts $h_{i,j}$ for $i \in [1,2]$ and $j \in [1,2]$, into the onion α which is encoded into a QR-code.
4. Generates the ballot, $s = (s^A, s^B)$ where, s^A =(Sigurd, Anna, Marit) and $s^B = \{250, \text{QR}\}$.



Voting phase Now, the voter makes her choice and votes for Anna, obviously. She separates the two parts: side A is thrown away while side B is scanned, and then kept for verification by the voter.

Decryption The two mix-servers decrypt $\alpha = \alpha_{2,2}$ and shift the place of the cross namely $\beta = \beta_{2,2} = 1$ to obtain the vote cast.

Mix-server $S_2 \in \mathcal{S}$ does the following:

1. Retrieves the pair $(\alpha_{2,2}, \beta_{2,2})$.
2. Decrypts $\alpha_{2,2}$ one layer, using his secret key $sk_{2,2}$, $D_{sk_{2,2}}(\alpha_{2,2}) = (t_{2,2}, \alpha_{2,1})$.
3. Computes the hash of $t_{2,2}$, $h_{2,2} = H(t_{2,2}) = 0 \bmod 3$.
4. Calculates the cyclic shift of the cross, $\beta_{2,1} = \beta_{2,2} - h_{2,2} = 1 - 0 = 1 \bmod 3$.
5. Obtains the pair $(\alpha_{2,1}, \beta_{2,1})$.
6. Decrypts $\alpha_{2,1}$ one layer, using his secret key $sk_{2,1}$, $D_{sk_{2,1}}(\alpha_{2,1}) = (t_{2,1}, \alpha_{1,2})$.
7. Computes the hash of $t_{2,1}$, $h_{2,1} = H(t_{2,1}) = 1 \bmod 3$.
8. Calculates the cyclic shift of the cross, $\beta_{1,2} = \beta_{2,1} - h_{2,1} = 0 \bmod 3$.
9. Obtains the pair $(\alpha_{1,2}, \beta_{1,2})$.

Mix-server $S_1 \in \mathcal{S}$ does the following:

1. Retrieves the pair $(\alpha_{1,2}, \beta_{1,2})$.
2. Decrypts $\alpha_{1,2}$ one layer, using his secret key $sk_{1,2}$, $D_{sk_{1,2}}(\alpha_{1,2}) = (t_{1,2}, \alpha_{1,1})$.
3. Computes the hash of $t_{i,2}$, $h_{i,2} = H(t_{i,2}) = 2 \bmod 3$.
4. Calculates the cyclic shift of the cross, $\beta_{1,1} = \beta_{1,2} - h_{1,2} = 0 - 2 = 1 \bmod 3$.

5. Obtains the pair $(\alpha_{1,1}, \beta_{1,1})$.

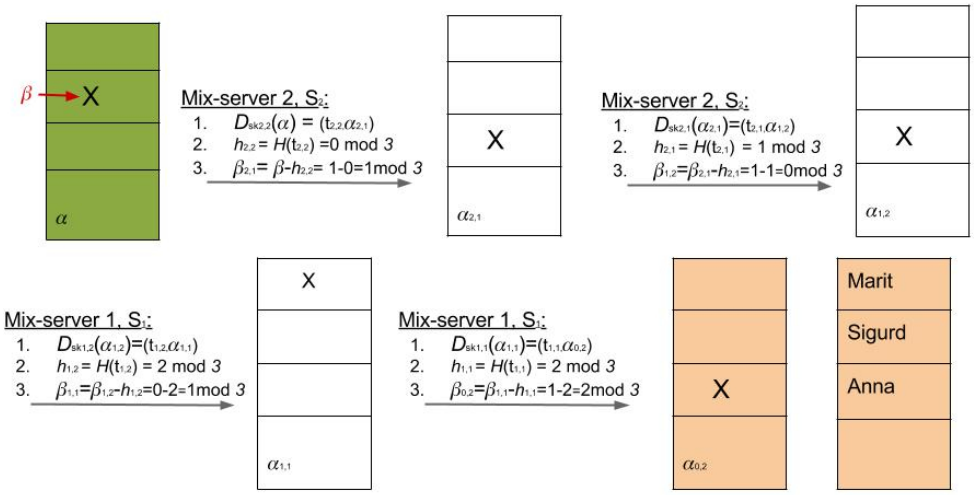
6. Decrypts $\alpha_{1,1}$ one layer, using his secret key $sk_{1,1}$, $D_{sk_{1,1}}(\alpha_{1,1}) = (t_{1,1}, \alpha_{0,2})$.

7. Computes the hash of $t_{1,1}$, $h_{1,1} = H(t_{1,1}) = 2 \bmod 3$.

8. Calculates the cyclic shift of the cross, $\beta_{0,2} = \beta_{1,1} - h_{1,1} = 1 - 2 = 2 \bmod 3$.

9. Obtains the pair $(\alpha_{0,2}, \beta_{0,2})$.

After four decryptions, $\alpha_{0,2} = 2$ is obtained, and the initial order of the candidate list, (Marit, Sigurd, Anna), can be calculated, and matched with the cross which is decrypted to be in position, $\beta_{0,2} = 2$. A vote for Anna is counted.



As illustrated the vote is decrypted correctly.

4.4 Cryptographic description by re-encryption mix-net

Re-encryption mix-net is the printed-on-demand version of Prêt-à-voter. The basic idea is that the voter prints her ballot just before voting so that the election authority does not have access to ballots before the voting phase. Because the ballots are printed on-demand, each ballot contains two QR-codes encoding the encryption of the order of the candidate list. The QR-code on side A is decrypted by the ballot printer at once, so that the voter receives a ballot with a candidate list. The QR-code on side B is decrypted by the tellers during the tallying of the votes.

In this description, we omit an example and some parts of the cryptography that are very similar to the decryption mix-net version of Prêt-à-voter and focus more on the difference between these versions.

4.4.1 Setup phase

There are five different types of agents involved in the re-encryption mix-net version of Prêt-à-voter: the election authority divided into groups, the mix-servers, the tellers, the bulletin board and the voters.

- The election authority includes the set of voters, the set of candidates, the set of mix-servers and the set of tellers in the voting system. The election authority divided, into groups, generates the ballot forms.
- The mix-servers shuffle and re-encrypt the received votes.
- The tellers generate key-pairs both for encryption of ballots and decryption of votes and publish the election result.
- The bulletin board passively provides storage of information used for verification and results.
- The voter votes for the candidate of her choice, and can verify that her vote was counted correctly.

For simplicity, we assume that the election authority divided into groups, the mix-servers and the tellers are all divided into k groups. In reality, this number can be different for each types of agents.

The election authority divided into groups, $\mathcal{G} = \{G_1, G_2, .., G_k\}$, determines the set of mix-servers $\mathcal{S} = \{S_1, S_2, ..., S_k\}$, the set of voters $\mathcal{V} = \{V_1, V_2, ..., V_l\}$, the set of candidates $\mathcal{T} = \{T_1, T_2, ..., T_m\}$ and the set of tellers $\mathcal{N} = \{N_1, N_2, ..., N_k\}$. The tellers and the ballot printer generate two independent keys using exponential ElGamal presented in chapter 2.1.2.

Key Generation using exponential ElGamal

We assume the ElGamal parameters (g, p, q) are made public in advance. p and q are large primes such that $q \mid p - 1$, and g is a generator of $\mathbb{Z}_q$ which is isomorphic to a subgroup of $\mathbb{Z}_p^*$ with order q.

- The ballot printer randomly selects a secret key $sk_r \in \mathbb{Z}_q$ and reveals its public key $h_r = g^{sk_r}$.
- The set of tellers $\mathcal{N}$ generates the secret key $sk_t \in \mathbb{Z}_q$ in a threshold fashion. Then publishes the corresponding public key $h_t = g^{sk_t}$.

Encryption The election authority divided into groups, $G_1, G_2, .., G_k$, is in charge of the encryption using the public keys. They have to encrypt the order of the candidate list in two ways. The first encryption, using h_r is decrypted later by the ballot printer during the voting phase. The second encryption using h_t is decrypted later by the set of tellers during the tallying phase.

1. G_1 randomly selects $\alpha_1 \in \mathbb{Z}_m$ and $x_1, y_1 \in \mathbb{Z}_q$ to generate an encryption pair

$$(g^{x_1}, {h_r}^{x_1} \cdot g^{-\alpha_1}) \text{ and } (g^{y_1}, h_t^{y_1} \cdot g^{-\alpha_1}).$$

2. For $i \in [2, k]$:

   - G_i randomly selects $\alpha_i' \in \mathbb{Z}_m$ and x_i', $y_i' \in \mathbb{Z}_q$ to generate an intermediate encryption pair

$$(g^{x_i'}, {h_r}^{x_i'} \cdot g^{-\alpha_i'}) \text{ and } (g^{y_i'}, h_t^{y_i'} \cdot g^{-\alpha_i'}).$$

   - G_i multiplies the intermediate onion pair with the encryption pair received from G_{i-1}

$$(g^{x_i}, {h_r}^{x_i} \cdot g^{-\alpha_i}) = (g^{x_i'}, {h_r}^{x_i'} \cdot g^{-\alpha_i'}) \cdot (g^{x_{i-1}}, {h_r}^{x_{i-1}} \cdot g^{-\alpha_{i-1}})$$

$$(g^{y_i}, {h_t}^{y_i} \cdot g^{-\alpha_i}) = (g^{y_i'}, {h_t}^{y_i'} \cdot g^{-\alpha_i'}) \cdot (g^{y_{i-1}}, {h_t}^{y_{i-1}} \cdot g^{-\alpha_{i-1}}).$$

3. The final encryption pair is: $(g^x, {h_r}^x \cdot g^{-\alpha})$ and $(g^y, {h_t}^y \cdot g^{-\alpha})$, where

$$x = x_k = x_1 + \Sigma_{j=2}^{k} x_j' \mod q$$

$$y = y_k = y_1 + \Sigma_{j=2}^{k} y_j' \mod q$$

$$\alpha = \alpha_k = \alpha_1 + \Sigma_{j=2}^{k} \alpha_j' \mod q.$$

The order of the candidate list written on the ballot is denoted by α. Note that α must be equal in both encryption.

Generating a ballot We now describe the process of generating a ballot. A ballot must include two sides, A and B. Side A contains a QR-code containing encrypted information of the list of the candidates in an alphabetical order $(T_1, T_2, ..., T_m)$ with a cyclic shift α. Side B contains the ballot number, boxes aligned with the candidate list and the QR-code containing the same information encrypted in a different way.

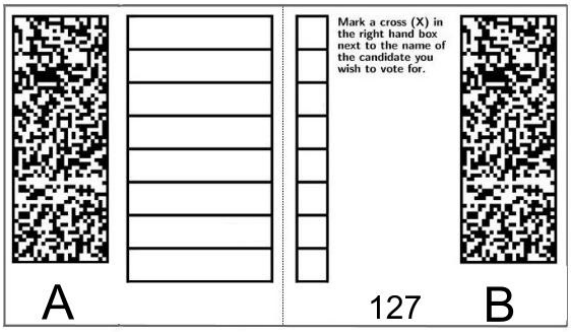
Generating a ballot

A ballot consists of side A, s^A, and side B, s^B. To create a ballot the election authority does the following:

1. Selects a unique ballot number, denoted *BN*.
2. Generates an order α of the candidate list obtaining $(T_1, T_2, ..., T_m)_\alpha$.
3. Encrypts the order of the candidate list α using h_r which is encoded into the QR_A-code on side A.
4. Encrypts the order of the candidate list α using h_t which is encoded into the QR_B-code on side B.
5. Generates the ballot, $s = (s^A, s^B)$ where, $s^A = \{\text{QR}_A\}$ and $s^B = \{BN, \text{QR}_B\}$.

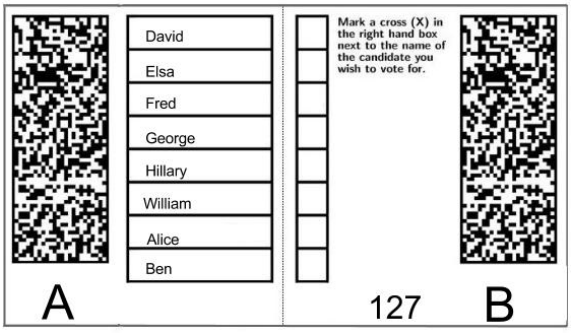
4.4.2 Voting phase

During the voting phase, the voter receives a ballot $s = (s^A, s^B)$.



A ballot before the ballot printer has printed on the candidate list.

She inserts the ballot into the printer. The ballot printer decrypts the code on side A, QR_A by using the secret key sk_r, and prints out the candidate list in the order $(T_1, T_2, ..., T_m)_\alpha$ corresponding to the decrypted code.



A ballot after the ballot printer has printed on the candidate list.

The voter chooses the candidate of her choice, $T_i \in \mathcal{T}$, and marks a cross in the corresponding box on s^B. The vote cast is $V_{\text{cast}} = (s^B, \beta)$ where β is the position of the cross. She destroys s^A and receives the receipt, $V_{\text{receipt}} = (s^B, \beta)$.

4.4.3 Tallying phase

After the voting phase, side B of all the ballots are published on the bulletin board.

Including the mark on the ballot in the encryption The election authority must include the marks on each ballot in the encrypted code $(g^y, {h_t}^y \cdot g^{-\alpha})$ on side B so that it can be tallied. The position of the cross, denoted by β, is included in the encryption in the following way,

$$(g^y, {h_t}^y \cdot g^{-\alpha} \cdot g^{\beta}) = (g^y, {h_t}^y \cdot g^{-\alpha+\beta}).$$

Mix-net In order to provide privacy, the encrypted votes $(g^y, {h_t}^y \cdot g^{-\alpha+\beta})$ are re-encrypted and shuffled in a mix-net by the mix-servers using permutation functions. This procedure is very similar to the decryption and shuffle phase in the decryption mix-net in chapter 4.2.3. However, instead of decrypting, the mix-servers re-encrypt the votes. These re-encryption do not change $-\alpha + \beta$.

Tallying Finally, the random sequence of re-encrypted votes are decrypted by the tellers using their secret key sk_t. After decryption, the tellers finally perform $-\alpha + \beta = \gamma$. The tellers obtain that the voter voted for candidate T_γ in the alphabetical ordered candidate list $(T_1, T_2, ..., T_m)$. The decrypted votes are then included in the tallying.

Chapter 5

Requirements in the voting phase of Demos

We have described two different paper-based electronic voting systems, Demos and Prêt-à-Voter. The objective of the following chapters is to analyse Demos. We explore how Demos defines and ensures a trustworthy system by examining the privacy, verifiability and availability properties of the system. First we state the requirements needed in the voting phase at the polling station in chapter 5 and then the requirements needed in the technical part of the system in chapter 6. In chapter 7 we present the analysis by the article, referred to above, of the verifiability based on a verifiability game.

The motivation behind this first chapter of the analysis is to get a closer look at how Demos can be implemented in a real-life election. A study of the voting phase is omitted in the article presenting Demos by Kiayas, Zacharias and Zhang [11] which is why we focus on the voting phase in this chapter. We note that, in Demos, the election authority controls the whole election. He is responsible for the setup phase, the generating, and the tallying of the votes. Consequently we often analyse Demos in the case where the election authority is the adversary. When needed, we also look at external adversaries.

We start by analysing which requirements are needed during the voting phase, at first with respect to the privacy and then to verifiability. In other words, how to organise the voting phase at the polling station so that it is certain that both the voter privacy and the verifiability of the vote are kept.

Assuming side A is used for voting and $i \in [1, m]$, then a particular cast vote is presented as $V_{\text{cast}} = (BN, s^A, C_i^A)$ and the vote receipt, received after scanning the ballot, is presented as $V_{\text{receipt}} = (BN, s^B, C_i^A)$ in Demos. The ballot number is denoted as *BN*, the chosen side to vote with is represented by s^A and C_i^A is the vote-code of the chosen candidate. For verification, the other side, s^B, is used. We state requirements needed during the voting phase, and justify their importance.

5.1 Privacy

Requirement 1 *The election authority has no control over how the ballots are distributed among the voters.*
Without this condition a malicious election authority can break the privacy of a voter by distributing specific ballots to specific voters. For example if the voter receives a ballot memorized by an adversary, submits the vote $V_{\text{cast}} = (BN, s^A, C_i^A)$, the adversary will know the corresponding candidate of the vote-code C_i^A cast.

This requirement demands the election authority to be split into independent parties. A possible solution to weaken the election authority's attack, could be to leave a number of ballots inside the voting booth, making the voter choose for herself which ballot to use for voting. However, this creates the possibility of a chain attack, where the voter now becomes the adversary. She can take with her a number of ballots which she demands other voters to use. The receipt printed $V_{\text{receipt}} = (BN, s^B, C_i^A)$ contains the ballot number, the verification side and the vote-code chosen. With this receipt, the adversary can obtain proof of whether the voter cooperated with her or not. In other words to prevent against coercing, the voter can not leave the polling station with an unused ballot.

A better solution may be to implement a ballot printer, similar to the one used in Prêt-à-Voter [13]. Here the ballots are printed randomly on the polling station, so that the election authority supposedly has less control over the distributing of ballots.

Requirement 2 *The voter is alone in the voting booth and no one can look over her shoulder observing what she votes.*
Without the conditions above, the voter privacy is trivially broken. Either by a malicious election authority or an adversarial voter looking over the shoulder of the voter while she votes. Demos does not explain thoroughly what happens to the voting side s^A of her ballot after it is cast. With the requirement above, the voter is alone in the voting booth, maybe having the opportunity to keep her voting side without anyone noticing. This can be used as a receipt of her vote which can be exploited by an adversary. Hence it is important that the voter can not leave the polling station with her voting side. A possible realistic solution in the implementation of the system could be to make the voting machine destroy the voting side of the ballot immediately after it is cast.

5.2 Verifiability

Requirement 1 *Every voter should have the possibility of verifying the ballots in the voting phase.*
With test ballots, the voter can on the polling station make sure that her ballot has been constructed correctly, and that the verifying procedure is proper. By scanning a ballot the voter can verify that the candidate and vote-code correspondences $s = \{(T_i, C_i)\}_{i \in [1,m]}$ are identical on the ballot and on the screen. Without this condition, a malicious election authority can get away with manipulating ballots, and the following attacks become more relevant.

Requirement 2 *The choice of which side the voter uses for voting is random.*
The key to the verifiability of Demos is the voters random choice of side A or side B in the voting phase. This randomness is not easily implemented in a real election. A possible solution would be to use a ballot printer, deciding randomly which side to vote with and which to verify with, s^i, $i \in \{a, b\}$ chosen randomly. However we can not be certain this printer is not controlled by an adversary. Another solution could be to implement a real coin-toss for every voter, where for example heads corresponds to s^a and tails to s^b. This solution affects the availability of the system since it requires that an election authority verifies that the coin toss actually happens.

Without the requirement above, the malicious election authority can analyse the behaviour of voters and produce attacks based on this analysis. For example, if both sides are presented to the voter at the polling station as side A and side B such that side A is on the left and side B on the right, one can assume that most uninterested honest voters vote with A and use B for verification. The election authority can take advantage of this and produce a fake side A, but a real side B. Moreover, the analysis can go further by assuming that there is a higher probability that voters that vote with ballot A are the same voters that does not verify their vote. Hence these votes can be manipulated with higher probability without being caught.

We present attacks made by the adversary in the printing phase where we assume the random choice of side A and B is left to the voter. These attacks are modified versions of the attacks presented by Kiayas, Zacharias and Zhang [11]. First we introduce some common assumptions to make the description of the attacks simpler.

- The adversary knows that the voter votes for Marit with side A.
- The adversary wants Anna to win, and tries to change the vote to her advantage.

The Modification attack The Modification attack is based on changing the candidate and vote-code correspondence. The adversary copies and manipulates a real ballot and gives this to the honest voter. When this ballot is cast by the honest voter, the voting system will recognize it as the original ballot. In other words, in the system it is the real ballot that will be tallied. The adversary manipulates the copied ballot by doing the following:

- Swap the corresponding vote-codes for Anna and Marit on side A.
- Let side B be the same as on the real vote.

When the honest voter votes for Marit, by using what looks like the corresponding vote-code, it will be counted as a vote for Anna in the voting system. When the honest voter verifies the ballot with side B, she will see that the vote-codes corresponds correctly to the candidates stated. When the honest voter verifies her vote, the system confirms her that the vote-code she cast will be tallied correctly. Hence she can not figure out that her vote has been manipulated.

The Clash attack The Clash attack is based on making fake ballots by copying one ballot. The adversary wants to attack θ honest voters who want to vote for Marit. We assume the adversary can inject votes on the bulletin board, and proceed as follows:

- Injects $\theta - 1$ votes for Anna on the bulletin board.
- Makes $\theta - 1$ copies of a real ballot all having BN=102. These are recognized in the system as the $\theta - 1$ votes the adversary injected.
- Distributes the θ ballots to the group of honest voters.

When the θ ballots are cast by the honest voters, the system will recognize $\theta - 1$ of them as the votes already cast by the adversary for Anna. One of the θ votes is not manipulated, hence when the honest voters verify their votes all with BN=102, the system will show the verification of this vote. The $\theta - 1$ voters will not figure out that their votes have been manipulated. The adversary has managed to change $\theta - 1$ votes in the election result.

5.3 Conclusion

Demos may have privacy weaknesses during the voting phase. We argue that the requirements stated above must be satisfied and practical implementations of these must be found. While not enough privacy for the voter during this phase breaks the privacy of the system, too much privacy opens possibility of vote-selling and coercing. Demos must take into consideration both problems in an election. Regarding the verifiability, the attacks above show how the implementation of a real coin-toss is crucial. With this randomness, the attacks can be detected with a higher probability. This is discussed more in chapter 7.

Chapter 6

Analyzing cryptographic components of Demos

In this chapter we analyse Demos by investigating the main cryptographic components of the system, namely the commitment scheme and the bulletin board. We want to find out whether the privacy and verifiability requirements needed are satisfied. In the traditional Norwegian voting system, the setup phase and the tallying phase are understandable to a regular voter. By using cryptographic components, Demos makes these phases more difficult to trust. It is therefore important that both the commitment scheme and the bulletin board works correctly since it can not be verified by a regular voter.

6.1 Commitment scheme

The beauty of the structure of Demos is that most of the computations of the voting system are done during the setup phase using a commitment scheme. In other words the election authority does not have much control after this phase. This might increase the trust of the voters towards Demos in comparison to other voting systems, for example like Prêt-à-voter which is dependent on computations by different agents after the voting phase. Of course this trust requires that the commitment scheme is correctly constructed, so that a malicious election authority can not cheat the voters. We therefore look at the properties of a commitment scheme, and what is required for it to be secure.

In Demos the election authority commits to the encoded candidates, denoted $c((l+1)^{i-1})$ and the vote-codes, denoted $c(C_i)$ for $i \in [1, m]$, and publishes these on a bulletin board. If the commitment scheme is secure, the election authority can not change the encoded candidates or the vote-codes in the commitments. At the same time a third party can not know the correspondences between the commitments and the encoded candidates or vote-codes.

In other words a secure commitment scheme contains two properties namely the hiding

and the binding property. We investigate these properties, using the notations from the general commitment scheme presented in chapter 2.2.4. Let k be the key length in the voting system. As k gets bigger the security of the system increases.

As a reminder, $\mathbb{G}$ is the group generated by g and is isomorphic to $\mathbb{Z}_q$, where q is a prime. $\mathbb{G}$ is a subgroup of a group defined by the elliptic curve parameters pk. A random element ω in $\mathbb{Z}_q$ is used to generate $h = g^{\omega}$. The commitment key is $ck = (pk, h)$. How the commitment key is generated is described more thoroughly in chapter 3.2.1.

Commitment scheme based on exponential ElGamal

For $m, t \in \mathbb{Z}_q$ where t is random,

$$Com_{ck}(m) = (c, d) = ((g^t, g^m \cdot h^t), t)$$

is the commitment/opening pair form of m.
The commitment/opening pair is homomorphic under operation,

$$Com_{ck}(m_1) \cdot Com_{ck}(m_2) = Com_{ck}(m_1 + m_2)$$

for $m_1, m_2 \in \mathcal{P}$.

6.1.1 Privacy

Hiding definition For any commitment, c, it is computationally hard for any polynomial bounded adversary $\mathcal{A}$, given two messages $m_0, m_1 \in \mathcal{P}$, to distinguish between their corresponding commitments. In other words, $c(m)$ reveals no information of the message m. It is given that the commitment key ck is generated by a trusted party.

A commitment scheme satisfies the hiding property if the probability of success for an adversary who tries to distinguish between two commitments subtracting $1/2$ is negligible. We subtract $1/2$ because this is the probability of guessing the message and commitment correspondences. A trusted party, denoted *TP*, generates the commitment key, where the size of its input is k. A coin toss is represented by $b \leftarrow \{0, 1\}$ and $neg(k)$ represents a negligible function in k, presented in chapter 2.2.5.

The hiding property

$$Pr\left[b = \tilde{b} \;\middle|\; \begin{array}{l} ck \leftarrow TP(1^k), (m_0, m_1) \leftarrow \mathcal{A}, b \leftarrow \{0,1\} \\ (c, d) \leftarrow Com_{ck}(m_b), \tilde{b} \leftarrow \mathcal{A} \end{array}\right] - \frac{1}{2} \leq neg(k)$$

Following the definition above, we investigate if the commitment scheme in Demos satisfies the hiding property.

In Demos the election authority randomly chooses t from $\mathbb{Z}_q$ and keeps this value secret. The commitment of a message m is then $c(m) = (g^t, g^m h^t)$. The hiding property of the commitment is satisfied since t is chosen randomly from $\mathbb{Z}_q$ implying that the commitment $c(m) = (g^t, g^m h^t)$ is a random element in the group $\mathbb{G} \times \mathbb{G}$. In other words, since t is

random and hidden, the adversary can not distinguish between the commitment of m or any other element in $\mathbb{G} \times \mathbb{G}$.

Decisional Diffie-Hellman Assumption It is also important to note that the Decisional Diffie-Hellman Assumption holds over $\mathbb{G}$ in Demos. If not, assuming c' is a commitment to a random group element in $\mathbb{G}$, an adversary knowing $c(m_0)$, $c(m_1)$ could distinguish between $c(m_0 m_1)$ and c'.

The Decisional Diffie-Hellman Assumption does not hold for any presentation of a cyclic group [4]. For example, one simple attack occurs in the group $\mathbb{Z}_p^*$ where p is a prime, and a, b, c random integers and g the generator of the group. By computing the Legendre symbol of g^a and g^b one can easily find the Legendre symbol of g^{ab} and distinguish between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c).

The commitment scheme in Demos uses a group parametrized by the elliptic curve parameters $pk = (p, a, b, g, q)$. In this group the Decisional Diffie-Hellman Assumption holds. Because of that the only possibility for an adversary to distinguish between (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) is by using the full discrete logarithm algorithms.

The generation of the commitment key In the definition above, one assumes that the commitment key, ck, is generated by a trusted third party. But in Demos it is the election authority that controls the setup phase, in other words generates the commitment key. Therefore, a malicious election authority can generate a commitment key, ck, in ways allowing him to cheat. In this case an attack on the hiding property is possible. Since the election authority generates the commitment key, he knows the secret ω in $ck = (pk, h) = (pk, g^\omega)$. We make a short example to explain the attack. First note, if the election authority is in control of every step in the commitment scheme, the hiding property is broken trivially. Hence in the attack below we assume the election authority generates the commitment key while an independent agent commits to the messages.

The set of encoded candidates is $\{(l+1)^{i-1}\}$, for $i \in [1, m]$, where m is the number of candidates and l the number of voters. Let $(l+1)^v$, where $v \in [0, m-1]$, be an encoded candidate of the election. The commitment to $(l+1)^v$ is calculated by an agent independent of the election authority such that $c((l+1)^v) = (g^t, h^t \cdot g^{(l+1)^v})$, where t is the random value hiding the message. The election authority knows all the encoded candidate selections in $\{(l+1)^{i-1}\}$, for $i \in [1, m]$, since the number of candidates is relatively small. By observing the commitment to $(l+1)^v$ the malicious election authority extracts the value g^t and then calculates $(g^t)^\omega$. By multiplying this value with each possible $g^{(l+1)^{i-1}}$ for all $i \in [1, m]$ he eventually finds an $i \in [1, m]$ such that

$$(g^t)^\omega \cdot g^{g^{(l+1)^{i-1}}} = h^t \cdot g^{(l+1)^v} = (g^\omega)^t \cdot g^{(l+1)^v}$$

$$\Rightarrow (l+1)^{i-1} = (l+1)^v.$$

The malicious election authority knows that the encoded candidate $(l+1)^v$ corresponds to the commitment $c((l+1)^v)$. With this attack the hiding property of the commitment scheme is broken, and hence the privacy requirement in the voting system is not fulfilled.

We discuss possible solutions of this problem, after introducing similar difficulties with the second property of a commitment scheme, namely the binding property.

6.1.2 Verifiability

Binding definition It is computationally hard for any polynomial bounded adversary $\mathcal{A}$ to find a collision in commitments, let's say (c, d_0) and (c, d_1) such that $Open_{ck}(c, d_0) \neq Open_{ck}(c, d_1)$ for two distinct messages m_0 and m_1. It is given that the commitment key ck is generated by a trusted party.

A commitment scheme satisfies the binding property if the probability of success for an adversary whose goal is to open a commitment in distinct ways is negligible. A trusted party, denoted *TP*, generates the commitment key, where the size of its input is k. A coin toss is represented by $b \leftarrow \{0, 1\}$ and $neg(k)$ represents a negligible function in k, presented in chapter 2.2.5.

The binding property

$$Pr\begin{bmatrix}(m_0 \neq m_1) \\ \wedge(m_0, m_1 \neq \perp)\end{bmatrix} \left| \begin{array}{l} ck \leftarrow TP(1^k), (c, d_0, d_1) \leftarrow \mathcal{A}, \\ m_0 \leftarrow Open_{ck}(c, d_0), m_1 \leftarrow Open_{ck}(c, d_1) \end{array} \right] \leq neg(k)$$

In Demos the binding property of the commitment scheme strengthens the verifiability of the voting system. When the commitment scheme is binding it is infeasible for a malicious election authority to open the committed message to another. For example in Demos if the commitment scheme is binding it is infeasible to change the encoded candidate or the vote-code after the commitment. Following the definition above, we investigate whether the commitment scheme in Demos satisfies the binding property.

Assume the adversary wants to open the commitment c in distinct ways. In other words he wants to find two opening pairs $(m_0, t_0), (m_1, t_1)$ where $m_0 \neq m_1 \bmod q$ and $m_0 = Open_{ck}(c, t_0)$, $m_1 = Open_{ck}(c, t_1)$. We get,

$$c(m_0) = c(m_1)$$

$$\Rightarrow (g^{t_0}, h^{t_0} g^{m_0}) = (g^{t_1}, h^{t_1} g^{m_1})$$

$$\Rightarrow (t_0 = t_1) \bmod q \Rightarrow g^{m_0} = g^{m_1} \Rightarrow m_0 = m_1 \bmod q.$$

The result contradicts our assumption where $m_0 \neq m_1 \bmod q$. Hence the adversary can not open the commitment in distinct ways which implies that the commitment scheme in Demos satisfies the binding property.

We note that the binding property still holds when the commitment to a message m is $c(m) = h^t g^m$.

$$c(m_0) = c(m_1) \Rightarrow g^{m_0} h^{t_0} = g^{m_1} h^{t_0} \Rightarrow h = g^{(m_1 - m_0)(t_0 - t_1)^{-1}}$$

Hence the adversary can directly calculate $(m_1 - m_0)(t_0 - t_1)^{-1} \bmod q$, which is the discrete logarithm of h, providing that $(t_0 - t_1)^{-1}$ exists. Since the Decisional Diffie Hellman assumption holds over $\mathbb{G}$ we have that discrete logarithms are infeasible to compute.

This contradicts the attack above, where we just found a method of computing the discrete logarithm of h.

The generation of the commitment key As we mentioned in the privacy section, the assumption that a trusted party generates the commitment key, may not hold in Demos. This time we assume that the election authority generates the commitment key and also commits to the messages, since this will not trivially break the binding property. Even with this control a malicious election authority can not break the binding property in the commitment scheme implemented in Demos. If $c(m_0) = c(m_1)$, $(g^{t_0}, h^{t_0}g^{m_0}) = (g^{t_1}, h^{t_1}g^{m_1})$ which implies that $m_0 = m_1$. In other words a malicious election authority can not open a commitment to distinct messages.

In the case where the commitment to a message m is $c(m) = h^t g^m$ and the Decisional Hellman Assumption does not hold, an adversary generating the commitment key can attack the binding property of the commitment scheme. Assume he generates $h = g^\omega$ such that $\omega = (m_1 - m_0) \cdot (t_0 - t_1)^{-1}$, where m_0 and m_1 are two distinct messages, and t_0, t_1 are their corresponding commitment openings. Then commitment c can be opened in distinct ways.

$$c(m_0) = c(m_1) \Rightarrow g^{m_0}h^{t_0} = g^{m_1}h^{t_1} \Rightarrow g^{m_0}g^{\omega \cdot t_0} = g^{m_1}g^{\omega \cdot t_1}$$

$$\Rightarrow m_0 + t_0(m_1 - m_0) \cdot (t_0 - t_1)^{-1} = m_1 + t_1(m_1 - m_0) \cdot (t_0 - t_1)^{-1}$$

$$\Rightarrow (m_1 - m_0) \cdot (t_0 - t_1)^{-1} = (m_1 - m_0) \cdot (t_0 - t_1)^{-1}$$

$$\Rightarrow Open_{ck}(c) = m_0 \vee Open_{ck}(c) = m_1$$

Proposal of new commitment key generation We present two solutions to prevent the attacks that occur when a malicious election authority has full control over the generation of the commitment key.

1. Let ck be generated in a threshold fashion among several independents parties.
2. Implement a hash function in the commitment key.

In the implementation of the first solution we need a threshold scheme presented in chapter 2.2.6. The solution requires that the election authority in Demos is split into several parts in the setup phase of the system, so that the generation of the secret, ω, of the commitment key $h = g^\omega$ is divided in a threshold manner. This could improve the privacy of the voting system and prevent the attack above, since the secret ω no longer is generated by only one agent. Splitting the election authority into different independent parties may not be simple in real life. One would have to be certain that the independent parties could not cooperate.

The second solution is to implement a hash function, H. In the following algorithm we find a new commitment key $h = (x, y)$. We assume the group structure is as defined in the commitment scheme of Demos, $a, b \in \mathbb{F}_p$ defining the elliptic curve $E(\mathbb{F}_p)$ by the equation, $E : y^2 = x^3 + ax + b \bmod p$.

Algorithm to generate h

1. $x_0 \leftarrow H(...)$.
2. $y_0 \leftarrow x_0^3 + ax_0 + b$.
3. if y_0 is a quadratic residue.
   (a) $x \leftarrow x_0$.
   (b) $y \leftarrow$ the smallest root of $\sqrt{(x^3 + ax + b)}$.
   (c) If $(x, y) \in \mathbb{G}$, output: (x, y).
4. else $x_0 \leftarrow x_0 + 1$ and return to step 2.

The election authority hash some random fixed values, for example the date, the number of voters etc, into x_0. Then by an algorithm he has to make sure x_0 is a point on the elliptic curve, and find $(x, y) \in E(\mathbb{F}_p)$.

By generating the commitment key using this algorithm, the privacy attack presented above is prevented. Now the election authority does not have any information of how h was generated, in other words no information of ω like in the key generation before.

Note that we assume that $\mathbb{G}$ is constructed so that it is a large subgroup of $E(\mathbb{F}_p)$, and that it is feasible to check whether elements are not in $\mathbb{G}$. In this way, 3(c) in the algorithm is easily computed.

Conclusion The commitment scheme plays a key role in the voting system Demos. By satisfying the hiding and the binding property, the commitment scheme strengthens the privacy and verifiability requirement of the voting system. After the election authority has committed to the encoded candidates and the vote-codes, it is infeasible for a third party to open the commitments and for anyone to change the commitments. The only weakness we find is that the election authority has full control over the setup phase of an election including the generation of the commitment key. A malicious election authority can take advantage of this power by breaking the hiding property of the scheme and consequently the privacy of a vote. On the other hand, even with this power, it seems that the malicious election authority can not break the binding property of the commitment scheme.

We also stress the importance of the election authority being divided into independent parts. If the agent committing to the vote-codes and to the encoded candidates is cooperating with the agent distributing the ballots at the polling station, the privacy of the vote can be broken. The agents can recognize the commitment to the vote-code marked by the voter.

6.2 Bulletin board

Demos, like many voting systems, depends on a bulletin board [9], publishing information and results of the election. We investigate what the requirements of a bulletin board in Demos are, and possible privacy and verifiability weaknesses surrounding this component.

First of all, the agents involved in the system of the bulletin board are the writer, the bulletin board and the reader. In Demos the writer is the election authority. He publishes keys, commitments and the tally of the votes. The reader is the election authority, voters, candidates or other neutral parties. They retrieve information and verify the voting system.

Before starting the analysis we point out that the bulletin board can not be controlled by the election authority. If this were not the case, a malicious election authority can manipulate the whole election breaking both the privacy and the verifiability of the system. The verifiability can be broken if for instance the malicious election authority replaces commitments on the bulletin board during the election. In other words the bulletin board needs to be an independent party.

Notation We introduce the following notation.

- The writers name is denoted W.
- The signature of the writer and the bulletin board are denoted s_W and s_B respectively.
- The time of when information has been transmitted by the writer and by the bulletin board are denoted t_W and t_B respectively.
- A hash function is denoted H.
- A hash with signature and time stamp by the writer and by the bulletin board are denoted $H.s_W.t_W$ and $H.s_B.t_B$ respectively.
- The transcript at the bulletin board available for the reader at time t is denoted τ_t.

6.2.1 Requirements

Now we identify three properties needed of the bulletin board in Demos. First of all, the information published on the bulletin board can not be removed, replaced or changed. Moreover, no other than the election authority can publish information on the board. Finally, the order of the information on the bulletin board must correspond to the order in which the election authority published the messages.
To achieve the first property the following requirement is needed.

Requirement 1 *The reader must verify that the transcript on the bulletin board has not been altered with.* Assuming the reader viewed the transcript on the bulletin board at time t_0, and later at time t_1, the reader should confirm that τ_{t_0} is a prefix of τ_{t_1}. Hence the only possible differences of the two transcripts is that the latter can obtain further information

applied by the writer in the time period $t_1 - t_0$. In Demos and in voting systems in general, independent parties should verify with short time intervals that the transcript has not been altered with. This requirement is important for the verifiability property of the voting system to hold. Without it, the bulletin board can manipulate the setup, the ballot commitments and the tallying of the election without being detected.

To achieve the second property the following requirements are needed.

Requirement 2 *The messages on the bulletin board must contain the writers name and the signature of the bulletin board.*
If the name of the writer, W, is not included, the bulletin board can publish messages on behalf of the election authority without the reader noticing. Additionally the election authority can not prove that the message was not applied by him.

If the signature of the bulletin board, s_B, is not included the reader can not be sure that the message applied by the election authority has been approved by the bulletin board. And if the election authority manages to publish a message without the approval of the bulletin board, the bulletin board can not prove this.

Requirement 3 *In the communication between the election authority and the bulletin board, both parts need to authenticate themselves.* For the election authority to be certain that he communicates with the bulletin board an visa versa, they both need to prove themselves by signatures. Here the assumption made above that signatures of messages can only be signed by their respective writers is important. Without the requirement an adversary can apply information to the bulletin board, or the election authority can give information to an adversary. This authentication can be implemented by a signature scheme [15].

To achieve the third property the following requirement is needed.

Requirement 4 *The time of when the election authority publishes the messages and when the bulletin board approves the messages must be included on the bulletin board.* If t_W is not included, the election authority can argue that it was published before and this way cheat in the election. Additionally the election authority does not have any proof if the bulletin board does not publish within the time limit.

If t_B is not included, the election authority can not report if the bulletin board does not publish the message in time. Conversely the bulletin board can not prove he has published the message from the election authority in time.

6.2.2 Protocol

Now the election authority wants to publish a message m on the bulletin board. We assume that the agents approve the messages from each other if the time-stamp is within ϵ time limit and the H_λ is the hash of the last posted message on the bulletin board. With all the requirements now stated, the following protocol is presented:

	Bulletin board		Election authority
1.	$H_\lambda.s_B.t_B$	$\longrightarrow$	
2.		$\longleftarrow$	$(m.s_W.t_W, H(m.s_W.t_W), H.s_W.t_W)$
3.	$H.s_W.t_W.s_B.t_B$	$\longrightarrow$	

In the first step, the bulletin board sends the hash of the last message posted on the bulletin board, with a signature and a time stamp. The election authority approves if the message is less than ϵ old.

In the second step the election authority sends the message, signed and with a time stamp and additionally, a hash of this, and at last a signed version of this hash. The bulletin board approves if $t_W - t_B < \epsilon$.

In the third step the bulletin board sends a signed and dated version of the hash sent by the election authority. The election authority approves if $t_B - t_W < \epsilon$. Additionally the election authority checks on the bulletin board that the message has been posted next in the sequence. If this does not occur or another message is published than the one intended, the election authority can produce m together with the signature of the bulletin board, to prove that it has been manipulated or deleted. If the signatures and hashes are correct and the messages were sent within the time-limit, the message $M =< m, t_W, W, H, H.s_W, H.s_W.s_B.t_B >$ where $H = H(m.t_W.s_W, H_\lambda)$ is posted on the bulletin board.

The transcript τ on the bulletin board after l published messages will be:

$$< M_1, M_2, ..., M_l >$$

where, $M_i =< m_i, t_{W(i)}, W_i, H_i, H_i.s_W, H_i.s_W.s_B.t_B >$ are in timely order, $H_i = H(m_i.t_{W(i)}.s_{W(i)}, H_{i-1})$, and $H_0 = 0$.

With the requirements fulfilled above, the bulletin board is consistent. Any reader can observe that the transcript τ_{t_0} at time t_0 is a prefix of the transcript τ_{t_1} viewed at a later time t_1.

We make one last remark, turning to the robustness of the bulletin board. Though it does not have the power to manipulate information posted by the writer, it has the power to abort the election, by shutting down the bulletin board. To avoid this, one solution would be to distribute the bulletin board among independent parts, organisations or servers. This would make the bulletin board more robust against sabotage attacks.

6.3 Conclusion

The technical parts of Demos are based on a commitment scheme and a consistent bulletin board. Having a closer look at the commitment scheme used in Demos, we have seen that it is both hiding and binding, given that the commitment key is generated by a trusted third

party. We presented new methods to generate the commitment key, preventing an attack of a malicious election authority.

Because of its key-role in the system, we have explored what is required of a bulletin board, and how the messages securely can be transmitted. Demos relies heavily on a consistent bulletin board to fulfill the verifiability property. It prevents a malicious election authority to take control over the bulletin board and change information already posted.

The purpose of these cryptographic components in Demos weakens if only one of them works correctly. As part of a voting system a commitment scheme and a bulletin board are dependent of each other. We trust the commitment scheme because the commitments are published on the bulletin board. If the bulletin board is not consistent, the point of a election authority committing to encoded candidates and vote-codes in the setup phase of the election is lost. An election authority can replace commitments later, without being caught. Without a hiding and binding commitment scheme, there is no point of a consistent bulletin board. For example, the election authority can break the verifiability of the system by opening a commitment on the board to another message.

Chapter 7

Demos as an end-to-end verifiable voting system

Demos is presented as an end-to-end verifiable electronic voting system. The main focus of the voting system is to fulfill the verifiability requirement. Hence, the discussion of this chapter is around this notion and how it is presented, based on the Demos article by Kiayas, Zacharias and Zhang [11].

7.1 Verifiability Game

The article of Demos defines the notion of verifiability by a verifiability game. We present a simplified version of this verifiability game. As before, the system below is presented for an election where the voter votes for one candidate only. The verifiability game is similar for an election where the voter votes for multiple candidates.

In Demos, the election authority is the main agent, responsible for the election and the generation and tallying of the ballots. Hence the verifiability game is based on the fact that the election authority is the adversary. A challenger is introduced as the opponent in the game, contributing with honest and successful votes in the election. The adversary wins if he changes the election result without being caught.

We repeat how the election authority encodes the candidate selection. The vote is encoded into an m-bit string, where m is the number of candidates. The bit in the i-th position is 1 if and only if candidate T_i is voted for. Thus, the output of the result algorithm is a vector in $\mathbb{Z}_+^m$.

Before explaining further, we present the notations and different elements of the game. We denote $\mathcal{V}$ the set of voters and $\bar{\mathcal{V}}$ the set of honest voters.

The vote extractor The vote extractor, denoted $\mathcal{E}$, is an algorithm that takes as input the set of all the cast votes, $\mathcal{V}_{\text{cast}}$, together with the set of receipts of the honest votes, $\bar{\mathcal{V}}_{\text{receipt}}$.

The algorithm extracts a set of adversarial and undefined votes $\mathcal{V}_{\text{fail}}$ from $\mathcal{V}_{\text{cast}}$.

$$\mathcal{E}(\mathcal{V}_{\text{cast}}, \bar{\mathcal{V}}_{\text{receipt}}) = \mathcal{V}_{\text{fail}} \quad \text{where,} \quad \mid \mathcal{V}_{\text{cast}} - \bar{\mathcal{V}}_{\text{receipt}} \mid = \mid \mathcal{V}_{\text{fail}} \mid$$

The purpose of the vote extractor algorithm is to report the dishonest and undefined votes in the result.

The deviation The goal of the malicious election authority in this game is to figure out how much the final results can be changed without being caught. The distance between the real result and the published result of the malicious election authority is called the deviation, denoted d, of the voting system.

To express the deviation formally, d is defined by a metric, here derived by the $1-$norm, $||\cdot||_1$ scaled to half.

$$d : \mathbb{Z}_+^m \times \mathbb{Z}_+^m \longrightarrow \mathbb{Z}$$

Assume the results $R = (x_1, x_2, ..., x_m)$ and $R' = (x'_1, x'_2, ..., x'_m)$ where x_j, x'_j for $j \in [1, m]$ are the number of votes for the candidate T_j in R and R' respectively. For an adversary to change the result from R to R', he must change,

$$d(R, R') = \frac{1}{2} \mid\mid R - R' \mid\mid_1 = \frac{1}{2} \sum_{j=1}^{m} \mid x_j - x'_j \mid$$

number of votes. The norm is scaled to half, because the maximum number of votes that can be changed to obtain this difference is half of the absolute difference of the votes.

Example Let $v_1 = (0, 0, 1,)$ and $v_2 = (1, 0, 0)$ be two votes.

$$\mid v_1 - v_2 \mid = 2$$

To change v_1 to v_2, only one vote is changed, hence we use d_1 as metric:

$$d_1(v_1, v_2) = \frac{2}{2} = 1$$

Further we present the verifiability game between the adversary, denoted $\mathcal{A}$, and the challenger, denoted $\mathcal{C}$, with the vote extractor, $\mathcal{E}$, and the deviation, d. As mentioned earlier, the adversary fully controls the election authority of the voting system. The minimum number of voters that the adversary must allow to vote honestly and terminate successfully is denoted by θ.

Verifiability Game

1. $\mathcal{A}$ does the following:
   - Selects voters, $\mathcal{V} = \{V_1, ..., V_l\}$ and candidates, $\mathcal{T} = \{T_1, ..., T_m\}$
   - For each voter $V_i \in \mathcal{V}$, chooses either of the following:
     - To corrupt the voter.
     - To allow $\mathcal{C}$ to play on its behalf, by providing him with the candidate selections of the honest voters.
   - Provides $\mathcal{C}$ with the receipt of all $V_i \in \bar{\mathcal{V}}$.
   - Posts the results on the bulletin board.
2. Verifiability Game$(m, l) = 1$ if:
   - All θ honest voters cast their votes successfully and audit the result successfully.
   - One of the following:
     - The extractor does not report all the dishonest and unsuccessful votes.

       $$d(R, R') > 0$$

     - The extractor reports dishonest votes that actually are honest.

       $$\mathcal{E}(V_{\text{cast}}, \bar{\mathcal{V}}_{\text{receipt}}) \leftarrow \bar{\bar{\mathcal{V}}} \subseteq \bar{\mathcal{V}} \qquad \text{where } \bar{\bar{\mathcal{V}}} \text{ is a subset of honest voters.}$$

$\mathcal{A}$ wins if Verifiability Game$(m, l) = 1$.

$\mathcal{C}$ wins if Verifiability Game$(m, l) = 0$.

By using this verifiability game, we present the mathematical definition of verifiability.

Verifiability Definition

Assume the existence of an extractor $\mathcal{E}$ and that at least θ honest voters voted successfully. For $l, m, d \in \mathbb{N}$ with $d > 0, 0 < \epsilon < 1, 0 < \theta < l$, the system supports verifiability with an error ϵ.

$$Pr[\text{Verifiability Game}(m, l) = 1] \leq \epsilon$$

Demos fulfills the verifiability requirement if the probability that the adversary changes the election result by a shift of d votes is less than or equal to an error ϵ.

7.2 The verifiability proof

The article proves that Demos is an end-to-end verifiable voting system by showing how different attacks including the modification and clash attack, presented earlier, will not succeed. It presents the weakness of the system by these attacks, showing that in Demos a malicious election authority can manipulate the ballot in the printing phase and how the vote is tallied. The article argues how the probability of a malicious election authority getting away with these attacks is negligible. As we have explained earlier, one side of the ballot will be used to vote with, the other to verify that the ballot is correctly constructed. Detecting that one of the two sides is manipulated is done by probability 1/2, because of the random choice of ballot side.

In other words the random coin-toss in the voting phase is the key reason of why Demos is end-to-end verifiable. In this chapter we briefly explain how the article proves the verifiability, focusing on the technical part depending on the voters coin toss. For interested readers we recommend further readings in the article where they present a σ-protocol of ballot correctness which is omitted in the thesis.

7.2.1 Producing the proof

As in the presentation of Demos, we assume a 1-out-of-m election, where the voter can only vote for one candidate.

The article of Demos introduce two types of attacks against the verifiability of the system. The first type is introduced by the modification and the clash attack, where vote-codes on the ballot corresponds to other candidates than what is printed on the ballot. The second type is that a malicious election authority can manipulate the vote for the encoded candidate $(l+1)^s$ for some $s \in [1, m]$ to be tallied as for example $1000 \cdot (l+1)^s$.

Given the attacks above, a conjunction of a cut-and-choose proof with a σ-protocol proof of a committed value belonging to a set, assures that the probability of a malicious election authority to not get caught is negligible.

The first attack against the verifiability is weakened by the cut-and-choose argument. Since we assume the voter verifies the ballot with the side that was not used to vote with, the probability of a malicious election authority getting caught is 1/2. The second attack is weakened by a σ-protocol proof of ballot correctness that provides assurance that the tallied commitment commits to one encoded candidate, but note not necessarily the correct one because that would break the privacy. To sum up, by a cut-and-choose argument, the article of Demos concludes that the election authority only needs to prove correctness with a σ-protocol of a single commitment because:

1. with the implemented coin toss the probability of a malicious election authority getting caught is $1/2$.

2. with a σ-protocol the election authority proves that the commitment of the vote will be counted as one vote for one of the candidates.

Further the article proves that the σ-protocol satisfies special soundness, and the special honest verifier zero-knowledge property. We look further into the verifiers challenge in this protocol, extracted from the voters coin.

7.2.2 Producing the verifiers challenge

The article introduces the definition of a min-entropy. This is used to measure how uncertain the probability distribution of the coin toss must be for the voting system to fulfill the verifiability requirement. We introduce the concept of entropy and a randomness extractor.

Entropy The entropy [5] is the measure of the unpredictability of a set of outcomes. Given a set of n events with probabilities $p_1, p_2, ..., p_n$, the entropy is the measure of change after given events. The measure is maximum if the events have equal probability, i.e. $p_i = \frac{1}{n}$ for all $i \in [1, n]$ and zero if $p_i = 1$ and $p_j = 0$ for all $j \neq i$. In other words if all the events happen with equal probability, the unpredictability is large and if the probabilities are different the uncertainty decreases. Greater uncertainty implies greater entropy. We define Shannon entropy, denoted H_1, below where X is the discrete random variable with the n events described above as possible outcomes.

$$H_1(X) = -\sum_{i=1}^{n} p_i log_2(p_i)$$

Example Given $n = 3$ with $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{4}$ and $p_3 = \frac{1}{4}$, the entropy of the discrete random value X with these events as possible outcomes is,

$$H_1(X) = -\sum_{i=1}^{3} p_i log_2(p_i) = -(\frac{1}{2} log_2(2^{-1}) + \frac{1}{4} log_2(2^{-2}) + \frac{1}{4} log_2(2^{-2})) = \frac{3}{2}$$

In Demos a version of entropy called min-entropy is used. Let X be the discrete random variable with possible outcomes $1, 2, ..., n$ and with respective probabilities $p_1, p_2, ..., p_n$, then the min-entropy, denoted H_∞, is defined in the following way,

$$H_\infty(X) = -log_2(max(p_i)) \quad i \in [1, n].$$

In other words a min-entropy is the largest number b such that all events occur with probability at most $1/2^b$. With the example above we would get the following min-entropy,

$$H_\infty(X) = -log_2(max(p_i)) = -log_2(2^{-1}) = 1 \quad i \in [1, 3].$$

Randomness extractor The purpose of a randomness extractor is to produce a highly random output from a weaker entropy source.

We present below a simplified version of how the article of Demos defines the verifiers challenge in the σ-protocol for ballot correctness.

In the σ-protocol the prover wants to prove that a commitment commits to one encoded candidate. The verifier provides the prover by a guessing probability which is extracted

by the voters coins denoted by $a = (a_1, a_2, ..., a_l) \in \{0,1\}^l$. The min-entropy of the coins is only contributed by the honest voters, since the coerced voters does not choose randomly their voting side. We find it realistic to assume that the honest voters coins are independent, as we assume that the coin toss happen separately for each voter. Demos states that if this guessing probability, D, satisfies $H_\infty(D) \geq b$, then the probability of the prover cheating the verifier by guessing is at most $1/2^b$.

An attack by a malicious election authority, presented in the article, can occur if the $a_i \in a$ are in the order the votes were cast. The election authority can organize the cast protocol of the election so that all the honest voters are put in a specific order, reducing the challenge space to be at most $\log_2 \theta$, where θ is the minimum number of honest voters. Then the probability of the prover cheating the verifier by guessing is $1/2^{log_2 \theta} = 1/\theta$. Such a level of entropy is insufficient in order to provide a small enough verifiability error. The verifiability error ϵ should drop exponentially with θ.

In other words, it is important that a malicious election authority can not control the order of the votes. A possible solution is to use a randomness extractor that distributes the votes in a random fashion into subsets. If every subset has a high enough entropy then the probability of the election authority cheating the verifier is small.

To sum up, the article of Demos depends on the randomness in the honest voters coin toss in their proof of ballot correctness. In the view of the availability aspect, the coin toss assumption in a real election can be unrealistic. Not only cheating can occur, but honest voters may inadvertently introduce bias into the coin flips. In a deeper analysis it would be interesting to make a pessimistic probability model, and compare it with a statistical analysis of actual sample of coin flips.

7.3 Conclusion

With a verifiability game, the notion verifiability is defined mathematically. This definition is used to show how the voting system Demos satisfies the requirement. A cut-and-choose argument is used, narrowing the proof down to a σ-protocol of ballot correctness. The random coin toss is important not only in the voting phase, but also in the technical parts of the proof by providing entropy for the verifiers challenge. We do emphasize that this was a short dive into how the verifiability notion in the system is defined and motivate the reader to explore further in the article by Kiayas, Zacharias and Zhang [11].

Chapter 8

Closing Remarks

The primary goal of this thesis was to investigate the potential of paper-based electronic voting systems. We chose to learn more about Demos and Prêt-à-voter, both appealing because they introduce verifiability.

Firstly, we wanted to understand how such systems can be constructed. The investigations required understanding in cryptography, knowledge of voting systems and awareness of the human parameter. Our own presentation of Demos and Prêt-à-Voter together with concrete examples was one of the main contribution of this thesis.

Secondly, the goal was to understand what is required of a voting system, both in the voting phase and in the cryptographic components used in the system. We analysed these in Demos with respect to the privacy and the verifiability. The availability was always taken into consideration. The goal was to motivate the reader to understand that making a voting system trustworthy is complex. We found that the commitment scheme in Demos is both hiding and binding and that if the election authority is divided into independent agents, the commitment scheme is trustworthy. We also concluded that Demos depends on both the commitment scheme and the bulletin board working correctly. If one of the components does not satisfy the requirements the other component is weakened. At last, we investigated how the notion of verifiability in Demos is defined and proved mathematically. We concluded by making a relation between the voting phase and the more technical parts of Demos by the coin-toss. We emphasized that a coin-toss is probably not realistic to implement in an election.

If there had been more hours in a day we would also have investigated definition and proof of privacy in Demos. Although this thesis did not analyse Prêt-à-Voter, understanding this voting system has been important when analysing Demos and has contributed to our perspective of paper-based electronic voting systems in general.

A personal goal was to get a deeper understanding in this field and to obtain an overall idea of how cryptography can be used in a real life situation. We also wanted to investigate the relation between the mathematics and the human parameter as both must be taken into

consideration in an election. Finally, we wanted to acquire the knowledge required to follow ongoing research in electronic voting system.

Bibliography

[1] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *Advances in Cryptology—EUROCRYPT 2002*, pages 83–107. Springer, 2002.

[2] Michael R Clarkson Stephen Chong Andrew and C Myers. Civitas: A secure remote voting system. 2007.

[3] David Bismark, James Heather, Roger Peel, Steve Schneider, Zhe Xia, and Peter YA Ryan. Experiences gained from the first pret a voter implementation. In *Requirements Engineering for e-Voting Systems (RE-VOTE), 2009 First International Workshop on*, pages 19–28. IEEE, 2010.

[4] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic number theory*, pages 48–63. Springer, 1998.

[5] Daniel RL Brown. Formally assessing cryptographic entropy. *IACR Cryptology ePrint Archive*, 2011:659, 2011.

[6] David Chaum, Peter YA Ryan, and Steve Schneider. *A practical voter-verifiable election scheme*. Springer, 2005.

[7] Nikos Chondros, Alex Delis, Dina Gavatha, Aggelos Kiayias, Charalampos Koutalakis, Ilias Nicolacopoulos, Lampros Paschos, Mema Roussopoulou, Giorge Sotirelis, Panos Stathopoulos, et al. Electronic voting systems–from theory to implementation. In *E-Democracy, Security, Privacy and Trust in a Digital World*, pages 113–122. Springer, 2014.

[8] Alex Delis, Konstantina Gavatha, Aggelos Kiayias, Charalampos Koutalakis, Elias Nikolakopoulos, Lampros Paschos, Mema Rousopoulou, Georgios Sotirellis, Panos Stathopoulos, Pavlos Vasilopoulos, et al. Pressing the button for european elections.

[9] James Heather and David Lundin. The append-only web bulletin board. In *Formal Aspects in Security and Trust*, pages 242–256. Springer, 2009.

[10] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 61–70. ACM, 2005.

[11] Aggelos Kiayias, Thomas Zacharias, and Bingsheng Zhang. End-to-end verifiable elections in the standard model. In *Advances in Cryptology-EUROCRYPT 2015*, pages 468–498. Springer, 2015.

[12] Peter YA Ryan. Prêt à voter with paillier encryption. *Mathematical and Computer Modelling*, 48(9):1646–1662, 2008.

[13] Peter YA Ryan, David Bismark, James A Heather, Steve A Schneider, and Zhe Xia. The prêt à voter verifiable election system. *IEEE transactions on information forensics and security*, 4(4):662–673, 2009.

[14] Peter YA Ryan and Steve A Schneider. *Prêt à voter with re-encryption mixes*. Springer, 2006.

[15] D.R. Stinson. *Cryptography: Theory and Practice, Third Edition*. Discrete Mathematics and Its Applications. Taylor & Francis, 2005.