Electronic Voting in the Standard Model

Thomas Briner

September 2003

Abstract

Electronic voting schemes that claim to satisfy the property of receipt-freeness usually need strong physical assumptions which are not available in real life. In this paper we present a protocol that achieves receipt-freeness in a threshold model without unrealistic assumptions. It is designed for large scale votes. It uses an existing type of untappable channels for the initialization of a vote but only usual internet connections for the voting phase. The untappable channels are needed only in order to achieve receipt-freeness but are not mandatory for all other properties.

The protocol achieves receipt-freeness by allowing each voter to convince a votebuyer who is willing to pay for a certain vote even though the voter casted an arbitrary vote. Even if the votebuyer is able to eavesdrop all channels between voters and authorities except for the untappable ones, it is indistinguishable for him whether or not the voter is telling the truth.

In case of coercion, a voter who is forced to cast a certain vote is still able to make sure that the vote will be considered invalid and therefore ignored by the authorities without giving the coercer the opportunity to figure it out. All these properties hold under the assumption that no authority cooperates with a votebuyer or blackmailer.

A dishonest authority is able to prevent a voter from casting a vote. This cannot be prevented but at least it will be detected that some irregularity has occurred. It is possible that the correctness of the result can be influenced by dishonest authorities, but in a context of a large scale vote, the level of overall correctness can still be judged by detecting the number of such irregularities and comparing it to the result.

Universal verifiability is not achieved with this protocol. The protocol is based on a threshold on the number of honest authorities. This is no loss compared to the protocols that claim to have the property of universal verifiability in theory as they need additional elements e.g. a kind of bulletin board that do not exist in real life. To implement this bulletin board it has to be simulated by the authorities and therefore depends on the honesty of those authorities too.

Contents

1 Introduction

Any voting protocol — whether electronic or not — has to achieve at least privacy for the voter, availability and a high level of correctness of the result.
A threat to be considered is votebuying and blackmailing. One of the differences between conventional and electronic voting is that votebuying is possible in a conventional context only for a small number of votes. There is no obvious and efficient way for a voter to prove that the vote he casted is the one the votebuyer is willing to pay for. In contrast to this, in the electronic context the voter is able to show all randomness he has used for the encryption and if needed any private information such as secret keys to convince a votebuyer of the vote he casted. All private information that is needed to convince a votebuyer will be called the receipt.
A votebuyer who has eavesdropped the vote which was sent to an authority can now simply compare that vote with the one he generates using the voter's randomness and his private information to verify if this vote is worth the money he is willing to pay. This process could be completely automated and the result of a vote could be strongly influenced by votebuying if not changed. Therefore receipt-freeness in our opinion is a mandatory property of an electronic voting scheme. The property of receipt-freeness was first introduced by Benaloh and Tuinstra [**?**].

To achieve receipt-freeness one has to make sure that no voter can construct a convincing receipt. Another way of describing the same property is to make sure that such a convincing receipt can always be obtained using some fake protocol, no matter whether or not the voter has casted the desired vote. In this case the receipt loses all its evidence and votebuying is no longer attractive.
There are major differences how receipt-freeness is defined. Will a voter be completely honest and always follow the protocol, especially if one tells him to erase the randomness to make sure he cannot construct his receipt, or might he be completely dishonest and act in an arbitrary way at every moment? In the second case this gives a much stronger definition of receipt-freeness. At what time is a voter allowed to interact with a votebuyer? During the whole vote or only at a certain time e.g. after having casted his vote? Is it enough to be able to construct a "receipt" using a fake algorithm that will be accepted with a non-negligible probability or is it necessary that such a receipt will always be accepted? And what are the assumptions concerning the cooperation of an authority and a votebuyer? Can receipt-freeness still be guaranteed even if an authority cooperates with a votebuyer, as long as the voter knows which one it is? All these questions have to be answered to determine the exact definition of receipt-freeness.

Blackmailing is possible only in small scale. Nevertheless for a single voter it might be a strong threat that should be prevented too by a protocol that claims to be receipt-free. A blackmailer can force a voter to cast a certain vote or prevent him from voting. The best thing a voting protocol could achieve is to recognize such a coerced vote and ignore it. If a blackmailer asks a votebuyer to prove that he has casted a certain vote it is necessary that the voter is able to generate a fake receipt that will always be considered correct by the coercer.

Since the introduction of receipt-freeness several protocols that claim to achieve receipt-freeness have been published. All these protocols have in common that they do not work safely without strong physical assumptions such as a voting booth or untappable channels.
The task of this semester project was to construct a protocol that works with realistic assumptions and provides a high level of security including receipt-freeness.

2 Approaches

In order to get an overview and a more precise view on receipt-freeness we started with the study of some publications concerning voting protocols in general and receipt-free ones in particular. Some of the ideas found during those studies are part of the proposed protocol.

2.1 Receipt-free Secret-Ballot Elections [?]

In this paper from Benaloh and Tuinstra the property of receipt-freeness was first introduced. They present a protocol for a receipt-free yes/no voting protocol that needs a voting booth as physical assumption. They define a voting booth as a place where one can read or record from any channel, but not write to any channel. In that paper they explain that the difference between privacy and receipt-freeness is in the point that the voting booth does not only *allow* to keep a vote secret but it *requires* that a vote stays secret which is the crucial point in discussing receipt-freeness.

2.2 Receipt-free Electronic Voting Schemes for Large Scale Elections [?]

The protocol presented in [**?**] is a modification of [**?**] that fixed some problems. The author claims that this protocol works in large scale, although strong physical assumptions are needed. He redefined the property of receipt-freeness based on the framework used in his protocol. To achieve receipt-freeness untappable channels in the first version and a voting booth in the second version are assumed. His definitions are quite different from the ones in [**?**], as he defines a voting booth as a physical apparatus in which a voter can interactively communicate with one party and the communication is perfectly secret to all other parties, which seems to be similar to other definitions of a bidirectional untappable channel.

2.3 Deniable Encryption [?]

The goal of the paper of Canetti, Dwork, Naor and Ostrovsky [**?**] is quite different from the other papers'. They show how to encrypt a message and send it to a receiver in a way that a sender can claim to have encrypted a different message and a third party is not able to decide whether this is true with a high probability. They achieve this goal using a trapdoor permutation and a hard-core predicate.

The property that a sender can claim to have sent something different than he did and it is detected only with a small probability seems to be very close to the property of receipt-freeness. By taking a closer look into the protocol in depth we discovered that the difference lies in the point that a voter who wants to sell his vote to a votebuyer has the possibility to commit himself to a way how to choose his randomness even before the vote takes place. By doing this he loses the possibility of encrypting his vote in a deniable way and a votebuyer could just ask for such a commitment, e.g. a seed for a pseudo-random generator, and therefore he can be sure that the voter did really cast the vote he claims.

It is a subtle difference that in the context of the deniable encryption a sender wants to get the deniability but in the context of receipt-freeness a voter could want to lose the deniability. To get deniability the voter must be able to keep his vote private, but to get receipt-freeness the voter must not be able to lose this privacy. Receipt-freeness seems to be a much stronger property than deniability. Therefore this protocol could not be transformed such that it could be useful for later use in the context of a receipt-free protocol.

2.4 Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting [?]

In this thesis several different approaches to a receipt-free voting protocol are proposed, e.g a protocol based on ballot shuffling or a protocol using a randomizer which re-encrypts the votes and proves the correctness of this re-encryption in a designated-verifier zero-knowledge proof[1]. The ballot-shuffling protocol achieves receipt-freeness in a threshold context, the protocol with the randomizer achieves even universal verifiability and receipt-freeness. For this property physical assumptions, i.e. untappable channels are assumed.

The thesis contains a framework with several protocols, concepts and "tools" we have used in some way in our protocol.

2.5 Vote Flipping

The main advantage of this idea is that the physical assumptions, in this case untappable channels are needed only in the set-up phase of a vote.

In the set-up phase the voter contacts the authority e.g. the local government. The authority presents two encryptions and proves publicly that one of them is an encryption of "yes" and the other an encryption of "no". These two votes are in a random order. Furthermore the authority has to prove to the voter which encryption contains which vote. This proof is done as a designated-verifier zero-knowledge proof as proposed in [**?**]. The voter signs both votes.

At the beginning of the vote the authority publishes both signed and encrypted votes on the bulletin board. Now the voter sends a "1" or a "2" in order to make the decision which of the two votes he wants to choose. As these votes were encrypted by the authority, she knows the votes of all voters.

A similar, but more efficient way is to choose "yes" as a "1" and "no" as a "-1". The authority presents only an encryption of either "yes" or "no". The voter signs this encrypted vote and the authority publishes it on the bulletin board. The voter only has to specify if he wants to invert the vote, which can be done with at least some homomorphic encryption functions such as ElGamal-like ([**?**], [**?**], [**?**]) or Pailler encryption ([**?**]). The attack that a coercer chooses randomly one of the two votes cannot be prevented and this random vote will be tallied as all the others. Detection is not possible.

Another open question is how to make the protocol more realistic in a way that the voter does not have to have physical contact at the beginning of every vote and if so, such an agreement should last for more than only a constant number of votes.

[1]The idea of a designated-verifier zero-knowledge proof is that a zero-knowledge proof is combined with a second one. To convince a verifier the proofer must know at least one of the two secrets, as it is an OR-combination of the two proofs. The second proof shows the knowledge of the verifier's secret key which is assumed impossible for anyone except for the verifier himself. If the combination of these two proofs is passed by the verifier to somebody else, it loses all evidence about the first proof because the fact that the verifier knows his secret key is trivial.

2.6 Permuting the Candidates

In order to blind an encrypted vote we have considered the possibility to permute the order of candidates[2] in a different way for every voter. The problems with this idea have been fixed and the permutation is a part of the proposal for a concrete implementation of the protocol presented in this paper.

One of these problems was that a randomization attack still had a linear probability to produce a valid vote even if a vote was chosen as a vector of a length that includes components that represent an invalid vote. This problem was fixed using the concept of authentication tags.

In order to convince a voter that his vote will be correctly tallied, the authorities would have to prove that the re-permuting was done correctly. If such a proof, even a designated-verifier proof, is sent over a tappable channel, the property of receipt-freeness is lost. To create a protocol without any unrealistic physical assumptions the voter can therefore not be sure that the re-permutation is done correctly and has to trust that a certain number of authorities act honestly.

3 Suggestion of a Practical Protocol

The protocol is based on a homomorphic voting protocol. In the first part we will introduce a basic version of the protocol. At the end we will improve the protocol and present an incremental version.

3.1 Overview

The most obvious way to construct an electronic voting protocol is as follows: A voter chooses his vote and encrypts it, sends his ballot to an authority which decrypts the ballot and tallies the vote if it is from an entitled voter. This setting is used by electronic voting protocols which use a single central server. The whole protocol is based on trust. The authority knows the votes of every single voter in plaintext. Security including the correctness of the result depends completely on the honesty of that authority. Receipt-freeness is given as long as the authority does not cooperate with a votebuyer.

In order to prevent the authority from knowing each voter's vote in plaintext, a homomorphic voting protocol could be used. The voter encrypts his vote using a homomorphic encryption scheme as it is presented in Section **??** and sends it to the authority. The authority does not need to decrypt it to be able to tally. The votes can be added using homomorphic addition and only the result will be decrypted. At no point in time a single vote is available in plaintext and therefore privacy is guaranteed. If there are several authorities that have to cooperate in order to decrypt a vote it is even impossible for a dishonest authority to attack a voter's privacy. Our protocol is based on such a homomorphic voting protocol with threshold security.

The property of universal verifiability can be achieved by using an additional device where every entitled voter can publish his vote, but no voter nor an authority can erase. In a homomorphic voting protocol each voter will publish his ballot with the encrypted vote on that device and is able to check whether the sum of encrypted votes is equal to the encrypted tally. To prove the correctness of the decryption a witness is published. The main problem about universal verifiability

[2] In order to keep the protocol as general as possible we assume a vote where a voter has the choice between several candidates. In our model a vote is valid iff only one candidate is chosen, i.e. a 1-out-of-L vote. A vote in which the only valid votes are "yes", "no" and abstention can be mapped directly to the candidate model.

is that it depends on the existence of a device where everybody can read and write on, but nobody can delete anything. As such a thing, usually called a bulletin board, does not exist, it has to be simulated by the authorities. Now the security depends on their honesty. Therefore no voting protocol will be able to achieve universal verifiability in a real world scenario.
Our protocol is a threshold protocol and is secure only if a certain number of authorities acts honestly. It does not provide universal verifiability as this would depend on the number of honest authorities too to simulate the bulletin board.

Next, the problem of votebuying has to be considered. To sell his vote a voter has to convince the votebuyer that the vote he casted is the one the votebuyer is willing to pay for. In order to do so the voter will open all secret and random parameters in the encryption. If a votebuyer is able to eavesdrop the channels between voters and authorities he can easily check whether this proof holds. To achieve receipt-freeness, additional physical assumptions such as untappable channels or a voting booth are needed which generally do not exist in real life.

We have chosen to guarantee receipt-freeness, i.e. to prevent a voter from losing his secrecy even if he wanted to, by blinding the encrypted vote in an information-theoretically secure way. The voter blinds his encrypted vote using a key he has received from the authorities. In order to make sure that a votebuyer does not know this key, it is sent over an unidirectional untappable channel which exists in real life — in an envelope by mail.
This untappable channel is used only at the beginning of each vote. The voter has to be sure that the key comes from the authorities, therefore the envelope is signed. The key itself is not signed in order to make it deniable. The voter sends his encrypted and blinded vote to the authorities which have to recover the vote by undoing the blinding and tally it as usual in a homomorphic voting protocol. There is no way for the authority to prove that the unblinding was done correctly and the encrypted vote is the one the voter did cast, without losing receipt-freeness.

As the vote a votebuyer could eavesdrop is blinded information-theoretically, the voter is able to claim for each vote he casted that he has voted for some other candidate. This is possible because the encrypted vote and the encrypted and blinded vote are statistically independent of each other. A vote can be claimed to contain any information by presenting a fake blinding key. This blinding key is obtained using a fake algorithm which has to be known to every voter. As the original key was sent in a deniable way, each voter may claim having received the fake key from the authorities and may “prove” to the votebuyer that the vote he casted was the one the votebuyer is willing to pay for. This blinding could be implemented using a one-time pad. In our protocol we have chosen a more convenient way, namely a permutation.

Another attack that is still possible in this setting is to randomize votes. A coercer could force a voter to cast a certain vote. Although the coercer is not able to discover who this vote will be counted for, as he will not know whether or not a voter gives him the correct blinding key, this attack could affect the correctness of the result. We prevent the randomization attack by adding a tag to the ballot which depends on the encrypted and blinded vote and the blinding key. If this tag is not valid the authorities will ignore the ballot.

3.2 Model

Entities We consider a model with N authorities $A_1 \dots A_N$. This set of authorities is denoted by $\mathcal{A}$. One authority called A_1 has a special role during the first part of the protocol, the initialization phase. A_1 is the initiator of the vote and wants to generate the votecall that will be accepted by all other authorities.
There are no special assumption about A_1 concerning honesty than about the other authorities. It might be convenient to choose A_1 as some representative of the government.
The set of entitled voters $V_1 \dots V_M$ is denoted by $\mathcal{V}$. The size of $\mathcal{V}$ is M.

Communication Between each voter and every authority there is a unidirectional reliable synchronous channel. Between the authorities secure channels are assumed. Over these secure channels messages can be broadcasted from one authority A_i to all other authorities using a broadcast protocol that ensures consensus as defined in [**?**].
For the set-up of the vote, unidirectional untappable channels from the authorities to the voters are assumed.

Key Infrastructure For the voting protocol, a public key infrastructure is needed to guarantee the authenticity of the ballots from the entitled voters. That means that to each voter and each authority a secret key and the corresponding public key is associated. The public keys must be published authentically. The secret keys can be used to decrypt or sign messages, the public keys to encrypt messages and to verify signatures. Each entity must know his own secret key and the public keys from all the others to be able to check the different messages for authenticity.

Adversaries There might be one or several votebuyers who are willing to pay a voter for casting a certain vote. Another type of adversaries are one or several blackmailers that coerce voters to act in a certain way e.g. to cast a certain vote or not to join the vote at all.
Among the set of authorities $\mathcal{A}$ there might be a number of dishonest ones. The set of dishonest authorities is denoted by $\mathcal{A}_{\text{bad}}$. A threshold t is defined by the number of authorities that are required to decrypt an encryption. The size of $\mathcal{A}_{\text{bad}}$ is bound by

$$\mid \mathcal{A}_{\text{bad}} \mid < t.$$

The number of honest authorities is therefore always greater or equal to $N - t$. To make sure that the honest authorities are always able to decrypt the tally together, we assume that

$$t < \tfrac{N}{2}.$$

No authority may cooperate with a votebuyer or blackmailer by assumption.
Among the entitled voters $V_1 \dots V_M \in \mathcal{V}$, an arbitrary number can be dishonest and deviate from the protocol in any way. A dishonest voter can interact with a votebuyer at any point of the protocol. These dishonest voters are members of the set $\mathcal{V}_{\text{bad}}$.

3.3 Building Blocks

In this section we present the definitions of the building blocks together with a concrete implementation which fulfills the demanded properties and are used as part of this protocol.

3.3.1 Encryption

In order to encrypt the votes we need an asymmetric homomorphic encryption scheme.

Specification We consider a probabilistic public-key encryption function:

$$E_Z : \mathbb{V} \times \mathbb{R} \to \mathbb{E}, (v, \alpha) \mapsto e.$$

Z denotes the public key corresponding to the secret key z that is shared among the authorities, $\mathbb{V}$, $\mathbb{R}$ and $\mathbb{E}$ are groups.

The decryption function is

$$D_Z : \mathbb{E} \to \mathbb{V}, e \mapsto v.$$

An encryption function is semantically secure if for any given encryption e and any two candidates v_1 and v_2, where one of them is the decryption of e, it is infeasible to determine which vote is contained in e with probability significantly higher than 0.5, unless the secret key is known [**?**].

An encryption scheme is homomorphic if

$$E_Z(v_1) \otimes E_Z(v_2) = E_Z(v_1 \oplus v_2),$$

where $\otimes$ is the operation in the group $\mathbb{E}$ and $\oplus$ is the operation in the group $\mathbb{V} \times \mathbb{R}$.

In order to generate the secret key z and the corresponding public key Z we need a setup-protocol where the key pair (z, Z) is constructed in a way each authority obtains a share z_i of z in a (t, N) threshold secret-sharing scheme and is publicly committed to this share.
The encryption scheme must provide threshold security such that for any set of less than t authorities, it has to be infeasible to decrypt an encryption.
For the decryption of the tally we need a decryption protocol that works with a shared secret key and that gives us a witness to verify that the result of the decryption is correct.

Implementation As encryption function the ElGamal-like encryption, as proposed by [**?**] and modified by [**?**] and [**?**] could be used. It works in a group $\mathbb{G}$, $|\mathbb{G}|$ is prime with two generators g and γ such that $\langle g \rangle = \langle \gamma \rangle = \mathbb{G}$. The encryption scheme is defined by.

$$E_Z(v_i, \alpha_i) = (g^{\alpha_i}, \gamma^{v_i} Z^{\alpha_i})$$

where Z is the public key according to the secret key z that is shared among the authorities.
This pair (z, Z) is generated in a setup protocol using a Shamir secret-sharing scheme with threshold t. After this protocol each authority A_i has a share z_i and is commonly committed to z_i by $Z_i = g^{z_i}$.
This encryption function is homomorphic, that is

$$E_Z(v_1, \alpha_1) \otimes E_Z(v_2, \alpha_2) = E_Z(v_1 + v_2, \alpha_1 \boxplus \alpha_2).$$

The decryption can be processed without reconstructing the secret key. Details can be found in [**?**]. To decrypt the tally needs time of $\mathcal{O}(\sqrt{\text{tally}})$ using the Baby-Step Giant-Step algorithm. As the tally is bound by the number of voters M, the decryption can be done efficiently.

Another possible implementation could be done using Paillier Encryption [**?**].

3.3.2 Deniable Blinding

In order to make receipt-freeness possible despite that only tappable channels are available at the moment a voter is casting his vote, the message — even an encrypted one — does not have to be sent as it is, but it needs to be blinded beforehand.

Specification A blinding function B takes a message m and a secret key b as input and maps m to $B_b(m)$. The knowledge of $B_b(m)$ gives no information about m as it is information theoretically independent of m. A deniable blinding function has the further property that for every message m, a key b' can be found efficiently such that ${B_{b'}}^{-1}(B_b(m)) = m'$ for an arbitrary m'.

Implementation One way to implement such a blinding function is to use a one-time pad as blinding key b and the xor-function as blinding function B. A blinded message $B_b(m)$ is information theoretically independent of the message m. It is evident that this implementation is deniable. In order to receive a key b' for a blinded message $B_b(m)$ and a different message m' we simply apply the xor-function to $B_b(m)$ and m'. A disadvantages of this implementation is the length of the secret key b that needs to be as long as the message m.

Other ways of implementing such a deniable blinding function can be chosen depending on the structure and the properties of the message. We will use a permutation function that permutes the order of candidates in our protocol because this implementation is very convenient for the chosen type of vote and gives a better performance.

3.3.3 Authentication Tag

The authentication tag T is used to express whether the voter wants a certain ballot to be counted or if the ballot – for example in case of coercion – should be invalid and ignored by the authorities. Together with the authentication tag T goes the symmetric authentication key $K_{\text{auth}_{\text{voterID}}}$ which is used to generate or verify an authentication tag T.

Specification In order to generate the authentication key $K_{\text{auth}_{\text{voterID}}}$ a key generation protocol is needed. This key generation is done by the initiator A_1. A possible implementation of this key generation protocol would be to choose randomly $K_{\text{auth}_{\text{voterID}}}$ uniformly distributed from the set of all possible authentication keys.
There are three protocols that are needed for this authentication tag:

- generation protocol, takes as input a message m and the authentication key $K_{\text{auth}_{\text{voterID}}}$ and generates the corresponding authentication tag T:
  $\text{generateTag}(K_{\text{auth}_{\text{voterID}}}, m) \rightarrow T$
- verification protocol:
  $\text{verifyTag}(K_{\text{auth}_{\text{voterID}}}, m, T) \rightarrow \{\text{true}, \text{false}\}$
- fake protocol:
  $\text{generateFakeKey}(\text{message } m, K_{\text{auth}_{\text{voterID}}}, \text{message } m') \rightarrow K'_{\text{auth}_{\text{voterID}}}$

  where message m' is the message that the voter claims to cast according to the votebuyer's desired vote.

  The probability distribution of $K_{\text{auth}_{\text{voterID}}}$, generated by the key generation protocol and of $K'_{\text{auth}_{\text{voterID}}}$ produced by the fake protocol have to be indistinguishable.

Implementation The following is one possible realization of the authentication tag T and the authentication key $K_{\text{auth}_{\text{voterID}}}$ using geometric constructs.
The authentication key $K_{\text{auth}_{\text{voterID}}} = (a, b)$ is a line[3], characterized by its equation

$$y = ax + b.$$

- generation protocol:
  To generate an authentication tag T, the generation protocol

  $$\text{generateTag}(K_{\text{auth}_{\text{voterID}}}, \text{message } m) \ \rightarrow \ T$$

  is realized as follows: The message m is the x-component, the authentication tag T is the y-component using the equation from the authentication key $K_{\text{auth}_{\text{voterID}}}$.

  $$T = a(\text{message } m) + b.$$

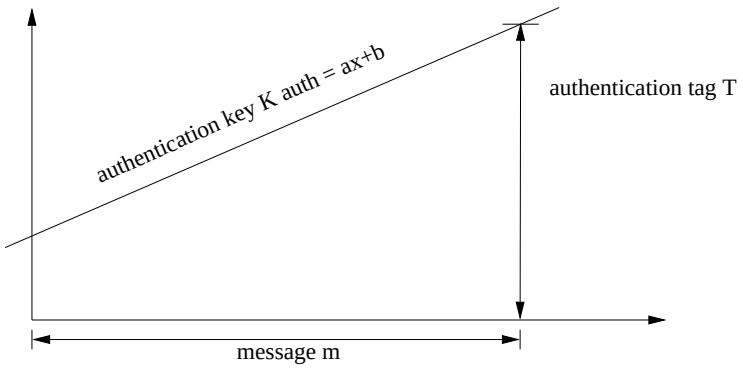


Figure 1: Geometric implementation of the authentication tag T

- verification protocol:
  To verify an authentication tag T with the verification protocol given the authentication key $K_{\text{auth}_{\text{voterID}}}$ and the message m, one just has to verify whether

  $$T = a(\text{message } m) + b$$

  is true or not.

[3]The reason for choosing two parameters randomly instead of only one e.g. using the set of all lines that run through the origin, is in a very special kind of attack. If a coercer knows a voter very well he could be able to guess which candidate the voter will vote for. Together with the ballot the voter casts this would give him full information about the authentication key $K_{\text{auth}_{\text{voterID}}}$. Therefore we have chosen the authentication key with two degrees of freedom.

- fake protocol:
  The most interesting part is the fake protocol:

$$\text{generateFakeKey}(\text{message } m, K_{\text{auth}_{\text{voterID}}}, \text{message } m') \rightarrow K'_{\text{auth}_{\text{voterID}}}.$$

  Given a message m and an authentication key $K_{\text{auth}_{\text{voterID}}}$, we first have to calculate the corresponding authentication tag T. Now we want to find another authentication key $K'_{\text{auth}_{\text{voterID}}}$ that fits together with this authentication tag T and the given message m'. In other words we have to find a line that goes through the point $P = (m', T)$. There are infinitely many so we can choose one randomly.

  The distribution of the fake authentication key $K'_{\text{auth}_{\text{voterID}}}$ is indistinguishable from the authentication key $K_{\text{auth}_{\text{voterID}}}$ which is produced by A_1.

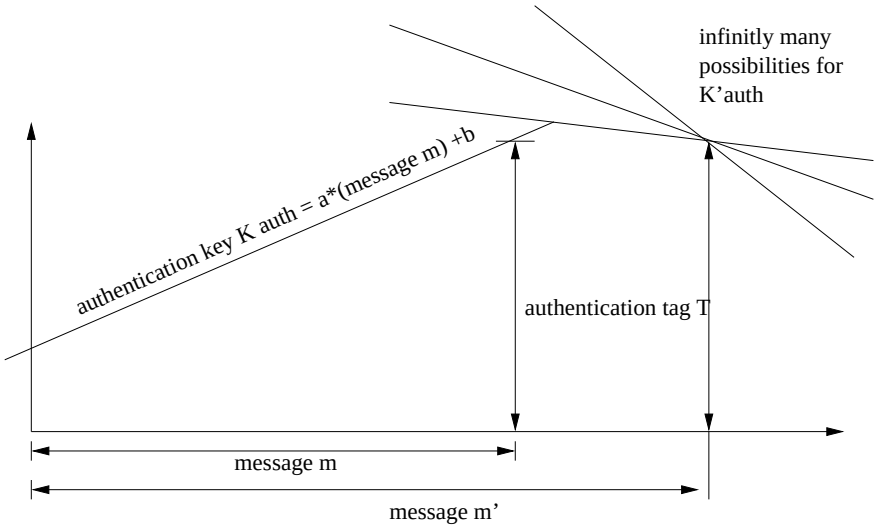


Figure 2: $K'_{\text{auth}_{\text{voterID}}}$ generated by the fake protocol

3.3.4 Authentic Deniable Transmission

To initiate this voting protocol the voter needs to receive some credentials i.e. in our protocol the secret key for the blinding function and the authentication key. The transmission of these credentials from an authority to a voter has to fulfill certain properties.

Specification The sender transmits a message m to a receiver. The receiver receives the message in an authentic way, such that he can be sure that the sender of this message is really the one who claims it to be. Nevertheless the receiver must not be able to convince anybody else that the message he presents is the one he has received from the sender. To an observer who is not able to read the transmission it is undecidable whether the receiver has produced the message by himself or whether he has received it from the sender.

Implementation There is no way to implement a transmission with the specified properties in the digital world. As soon as a message is transmitted authentically e.g. using some kind of signature, the receiver is always able to hand over the message together with the proof of authenticity to convince any third party that this message came from the sender.

But in the real physical world there are ways to implement such a transmission. It can be realized as a letter with the credentials on in an envelope which is sent by conventional mail. The envelope is signed by a conventional signature[4] from the sender. The receiver is able to check the correctness of that signature, as it is usually done in the real world.

The letter that is sent in this closed envelope has no letter-head or signature on it. There are only the credentials printed on this paper in such a way that anyone could produce a similar sheet of paper by himself[5].

The receiver who checks the signature, decides whether the signature is correct, opens the envelope and can be sure that the credentials come from the sender and that nobody was able to read them on their way from the sender to the receiver, as long as he trusts the mail company that this letter was not opened and the content was not manipulated. These assumptions of trust make sense in a large scale setting. The receiver is not able to convince anyone that he has received this letter as there is no evidence that he did not produce it by himself. Even if he hands over the envelope, nobody can be sure whether the envelope was opened before, the content exchanged and the envelope closed again by the receiver.

3.4 Basic Version of the Protocol

In this first version the protocol is divided into eight sequential phases. They follow each other in a synchronous way such that the next phase will not start before the previous phase is terminated. This division is chosen for reasons of comprehensibility. In the incremental version, which follows this basic version, some phases will be merged for reasons of performance.

1. The vote is initiated by A_1, the initiator. She generates the votecall vc that contains all information about the current vote. That means that she decides on the identifier for the current vote voteID, the list and the order of candidates, the list of entitled voters including their public keys, the list of authorities with their public keys and a random function $\mathrm{rf}_{\mathrm{seed}}$. This random function maps each voter to one authority. $\mathrm{rf}_{\mathrm{seed}} : \mathbb{V} \rightarrow \mathbb{A}$. A_1 has to prove that this function is chosen randomly. If all authorities are willing to participate in this vote and accept the votecall, they generate jointly a (z, Z) pair as specified in Section **??**. The public key Z is included in the votecall which is now signed by all authorities. If an authority does not sign A_1 has to modify the votecall and start over again. This votecall is published authentically.

   A_1 chooses randomly for every entitled voter an authentication key $K_{\mathrm{auth}_{\mathrm{voterID}}}$ which will be used by the voter to create the authentication tag with the properties described in Section **??** and a candidate permutation π_{voterID}, $\pi_{\mathrm{voterID}} : \{1, 2, .., L\} \rightarrow \{1, 2, .., L\}$. This permutation is the secret key for the blinding function introduced in Section **??** which is implemented as a permutation. A_1 broadcasts to all authorities the list of tuples $(\mathrm{voterID}, K_{\mathrm{auth}_{\mathrm{voterID}}}, \pi_{\mathrm{voterID}})$ for all voters. Each authority stores this list.

2. Each authority distributes $K_{\mathrm{auth}_{\mathrm{voterID}}}$ and π_{voterID} to those voters she is assigned to by $\mathrm{rf}_{\mathrm{seed}}$. This is done using the authentic deniable transmission as described in Section **??**. The voter has to be sure that he receives this information authentically from his authority. But he must not be able to convince anybody else that these are the $K_{\mathrm{auth}_{\mathrm{voterID}}}$ and π_{voterID} he has received from his corresponding authority. Therefore the authority signs the envelope

[4] An alternative way would be a digital signature of the receiver's address which is printed on the envelope. The receiver would have to type it in to check its correctness.

[5] In order to make sure that this is possible the government could even provide such a tool.

in which she sends these informations but she does not sign $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} itself in order to make them deniable.

If a voter does not receive an envelope with these informations or something with the signature or the content is not as it is supposed to be, he contacts the local government to get his $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID}.

3. The voter chooses his vote v by generating the vector with a "1" at the position of his desired candidate and a "0" everywhere else and encrypts it componentwise according to the encryption scheme from Section **??**. Then he applies the blinding function, i.e. his candidate permutation π_{voterID} to his encrypted vote e. In order to prove that this is an encryption of a valid vote he generates a validity proof that shows that all components are either "0" or "1" and that all components sum up to 1.[6]

   According to Section **??** the whole message should be blinded. That means that not only the encrypted message is blinded, but also the validity proof. Otherwise the proof could give information about the vote itself. We have decided to solve this problem by performing the validity proof to the encrypted and blinded vote. As the voterID and the voteID do not contain any information about the chosen vote, they remain unblinded. We will denote the encrypted and blinded message by $\tilde{e}$, the validity proof upon the encrypted and blinded vote by $\tilde{P}$.

   In order to obtain a valid authentication tag T for his vote, he uses the generation protocol $\text{generateTag}(K_{\text{auth}_{\text{voterID}}}, m) \rightarrow T$ from Section **??**. As message he uses the concatenation of the encrypted and permuted vote $\tilde{e}$ and his candidate permutation π_{voterID}. The ballot b is now simply a concatenation of all these parts: The voter's ID voterID, the current vote ID voteID, the authentication tag T and the validity proof and the signature upon all these components:

$$b = (\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P}, \text{sign}_{\text{SK}_{\text{voterID}}}(\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P})).$$

   In order to make sure that the ballot will be counted the voter sends it to at least one honest authority.

4. If the voter wants to sell his vote to one or several votebuyers he generates a "fake" authentication key $K'_{\text{auth}_{\text{voterID}}}$ for each vote he wants to claim having casted. These "fake" authentication keys are generated using the fake protocol from Section **??**:
   $\text{generateFakeKey}((\tilde{e}, \pi_{\text{voterID}}), K_{\text{auth}_{\text{voterID}}}, (\tilde{e}, \pi'_{\text{voterID}})) \rightarrow K'_{\text{auth}_{\text{voterID}}}$.

5. Each authority checks all received ballots whether the signature is correct and the ballot therefore is authentic from the entitled voter who's voterID is included in the ballot. If it is, she stores it, otherwise she ignores it.

6. At the end of the vote each authority broadcasts the stored votes and stores all the incoming ones from the other authorities. After this exchange all honest authorities will have exactly the same ballots stored.

7. All ballots are now checked for correctness and validity.

   A ballot is called *correct*, if the signature is correct corresponding to the voters public key, the voteID is the one of the current vote and the validity proof is correct.

[6] The first part is an AND-combination of L OR-proofs, the second part is straight forward using homomorphic addition.

A ballot is called *valid*, if the authentication tag T is correct. This is the case, if the verification algorithm verifyTag($K_{\text{auth}_{\text{voterID}}}, (\tilde{e}, \pi_{\text{voterID}}), T$) returns true.

In order to count the votes each authority has to distinguish between these different cases for all received ballots from one voter:

- If a ballot is incorrect either because of a wrong signature, an invalid proof or a wrong voteID it is ignored.
- If there are ballots which are correct but only invalid ones, she increases the number of accusations for the authority who has sent the envelope to that voter by one and ignores the ballot.
- If there is exactly one correct and valid ballot from a voter, it is added to the intermediate result by homomorphic addition.
- If there are several different correct and valid ballots from the same voter they are ignored.

The handling of the accusations is a political, not a technical concern. Possible treatments will be discussed in the summary in Section **??**.

8. The authorities decrypt the tally using the distributed decryption protocol from Section **??**. Each authority publishes the decrypted result together with the witness for the correctness of the decryption.

3.4.1 Initialization Phase

In the initialization phase the set of participating authorities $A_1 \dots A_N$ is determined and the parameters for the current vote are chosen, such as the voteID or the random function rf_{seed} to map the voters to the authorities.
The initiator A_1 also chooses the candidate permutation π_{voterID} and the authentication key $K_{\text{auth}_{\text{voterID}}}$ randomly for every entitled voter and broadcasts this information to all other authorities over the secure channels.

1. A_1 generates a votecall vc consisting of the following points:
   - voteID, identifying the current vote
   - list of candidates
   - order of candidates
   - list of participating authorities, each with her corresponding public key
   - list of entitled voters, including their public keys
   - random function rf_{seed} to map each voter to a corresponding authority based on the voters public key and a witness that rf_{seed} is chosen randomly, e.g. a hash of the votecall is used as seed.
2. A_1 asks all authorities to participate in the set-up protocol to generate the secret key z, which is shared among the authorities and the corresponding public keys as defined in Section **??**. The corresponding public key Z will be used to encrypt the votes by the voters.
3. A_1 includes the common public key Z to the votecall.

4. A_1 asks all authorities to sign the new votecall.

5. If one of the authorities does not accept to sign the votecall, the whole votecall has to be regenerated e.g. by correcting the complained points or by excluding this authority from the list of authorities.

   At this point the set of N authorities $A_1 \dots A_N$ is defined.

6. A_1 chooses an authentication key $K_{\text{auth}_{\text{voterID}}}$ randomly for every entitled voter as described in Section **??** and a candidate permutation π_{voterID} as secret key for the blinding function according to Section **??**.

7. A_1 broadcasts the list of generated tuples (voterID, authentication key $K_{\text{auth}_{\text{voterID}}}$, candidate permutation π_{voterID}), one for each entitled voter, to all authorities.

8. Each authority stores $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} for every voter.

Initiator A_1		**Authorities** $A_1 \dots A_N$
generates $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} for each voter.		
$\text{tuple}_{\text{voterID}} = (\text{voterID}, K_{\text{auth}_{\text{voterID}}}, \pi_{\text{voterID}})$	list of tuples →	store $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} for each voter

Analysis As the initiator is the one who sets the whole protocol in motion, it is his decision to start a vote. If he does not act according to the protocol and for example generates an incomplete votecall or does not choose the seed for the random function rf_{seed} randomly in Step 1., the vote will not take place as the authorities do not sign the votecall, and that is what the initiator could decide by his own anyway. Any misbehavior of the initiator is no stronger than what he is able to do anyway in his role as initiator.
If an authority does not behave correctly in Step 2., it will be excluded and will not take part in the vote. As each authority can decide in Step 5. whether she wants to sign or not, that is exactly what she could do anyway. If this authority misbehaves, she must be $\in \mathcal{A}_{\text{bad}}$ and therefore the invariant $\mid \mathcal{A}_{\text{bad}} \mid < t$ still holds after excluding the authority.
If the initiator chooses $K_{\text{auth}_{\text{voterID}}}$ or π_{voterID} in Step 7. in some deterministic way such that a votebuyer could be able to guess one or both of them, this is considered as a kind of cooperation, which is excluded by assumption. Due to the properties of the used broadcast protocol as they are defined in the model, it is impossible that some authorities could receive different information than others in Step 8.
If an authority does not store the received information in Step 8., she must be a member of $\mathcal{A}_{\text{bad}}$ as she does not follow the protocol. As a member of $\mathcal{A}_{\text{bad}}$ she can misbehave in any way, which means that she can for example invent a $K_{\text{auth}_{\text{voterID}}}$ or π_{voterID} as she likes.
By assumption we know that threshold $\mid \mathcal{A}_{\text{bad}} \mid < t$ and $t < \frac{N}{2}$ and therefore the majority of the authorities will have stored the correct information which will be necessary to check the validity of a ballot later in the protocol. Therefore this misbehavior does not influence the result of the vote.

3.4.2 Distribution Phase

In the distribution phase, the permutation π_{voterID} and the authentication key $K_{\text{auth}_{\text{voterID}}}$ are transmitted from the authorities to the assigned voters. This is done using the implementation of an authentic deniable transmission with conventional letters as described in Section **??**.

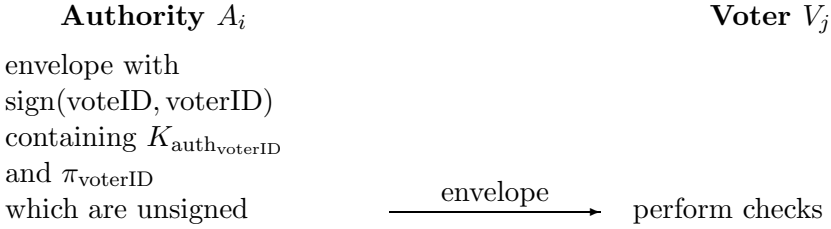
1. Each authority sends to the assigned voters an envelope with the ID of the current vote voteID, and the voters ID voterID, which are signed by the authority. This envelope contains $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID}, which are unsigned.

2. The voter receives this envelope and has to do several checks:

   He checks that:

   - it is the right authority to send him this envelope according to the random function rf_{seed} published in the votecall.
   - the signature on the envelope is correct.
   - the voteID is the one for the current vote.
   - the candidate permutation π_{voterID} and the authentication key $K_{\text{auth}_{\text{voterID}}}$ are of a valid size and type.

   If one check fails, the envelope is ignored.



3. If a voter does not receive an envelope or one of the checks fails, he has to contact the local government. There he receives the authentication key $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} authentically.

Analysis In Step 1. an authority could misbehave in many different ways: she could send nothing at all, or she could sign the envelope wrongly or use a wrong voterID or voteID, she could send an empty envelope or the envelope could contain wrong information in such a way that the voter can recognize that this cannot be the correct $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID}. The voter usually cannot prove this, as in case that a voter complains, there are both cases possible: Either the voter or the authority is dishonest. In both cases the voter can get his $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} at the local government. Of course he cannot be sure that these are the correct ones, as this authority could be a member of $\mathcal{A}_{\text{bad}}$ as well, but he can complain until he receives something that has at least the correct format, such that the voter cannot decide whether these are the correct informations.

If the authority sends a $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} which are of correct format but different from the ones the initiator has broadcasted, the voter will not be able to cast a valid ballot. This cannot be avoided but will at least be detected. If a voter casts correct but only invalid ballots, the number of accusations for this authority will increase by one. This can be the case if either the voter is dishonest or he is coerced or he has received wrong information. In the first two cases, these accusation will be more or less distributed equally over all authorities and therefore only give information about how smooth the vote was in general. In the third case, if an authority sends wrong $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} to many different voters there will be an accumulation of accusations for a certain authority. This will give important information for further votes. The result of a vote can be considered correct, if the total amount of accusations is negligible compared to the amounts of votes for the different candidates.

If anybody else, e.g. an authority the voter is not assigned to by the random function $\mathrm{rf}_{\mathrm{seed}}$, or a votebuyer or any third party sends an envelope with some wrong $K_{\mathrm{auth}_{\mathrm{voterID}}}$ and π_{voterID}, the voter will detect this by checking the signature and ignore this envelope.
If a voter is dishonest and claims not to have received the envelope or to have received it containing wrong data, this will not harm anybody.
If a voter has not received the envelope or with wrong data and the voter does not contact the local government, he will not be able to cast a valid ballot. That is what a voter can do anyway and will not change the result in any way. As the dishonest voters will be equally distributed over all authorities, the accusations for the authority that will be caused by such a ballot will be distributed equally too and have no further effect.

3.4.3 Voting Phase

To achieve both purposes — privacy and receipt freeness — the voter uses now the homomorphic encryption scheme and the received permutation as blinding function to encrypt and blind his vote. A vote v is a vector of length L, $v = (c_1, .., c_L)$. Each component stands for a candidate. A vote is valid if it has a "1" at the position of the candidate the voter wants to vote for and a "0" everywhere else. A "yes/no" voting is just a special case of this protocol where $L = 2$.
In Step 7. the voter casts his ballot that contains his permuted and encrypted vote and the authentication tag to mark any ballot as invalid that does not express his free will. Furthermore the voter has to prove that his vote is valid. This validity proof is performed upon the encrypted and blinded vote to guarantee that it contains no information about the vote. In order to make sure that the vote is authentic, the voter signs all information with the secret key $\mathrm{SK}_{\mathrm{voterID}}$ belonging to the $\mathrm{PK}_{\mathrm{voterID}}$ that is known to all authorities.

1. The voter chooses his vote v as a vector of length L with a single "1" at the position of the candidate he wants to vote for and a "0" everywhere else.

2. The voter encrypts each component of his vote using a homomorphic encryption function as defined in Section **??** with the random parameter chosen randomly and the authorities' public key Z.

$$e = E(v) = (E(v_1, \alpha_1), .., E(v_L, \alpha_L)),\ \alpha_i \text{ chosen randomly.}$$

3. As blinding function B the voter permutes his encrypted vote e using π_{voterID} he has received from the corresponding authority during the distribution phase. The encrypted and permuted vote is denoted by $\tilde{e} = (E(v_{\pi_{\mathrm{voterID}}(1)}, \alpha_{\pi_{\mathrm{voterID}}(1)}), \ldots, (E(v_{\pi_{\mathrm{voterID}}(L)}, \alpha_{\pi_{\mathrm{voterID}}(L)}))$.

4. The voter generates the authentication tag T using the generation protocol. The first parameter is the symmetric authentication key $K_{\mathrm{auth}_{\mathrm{voterID}}}$. The second parameter is the encrypted and permuted vote $\tilde{e}$ concatenated with his permutation π_{voterID}.

$$T = \mathrm{generateTag}(K_{\mathrm{auth}_{\mathrm{voterID}}}, (\tilde{e}, \pi_{\mathrm{voterID}}))$$

5. The voter generates the validity proof $\tilde{P}$ based upon the encrypted and blinded vote. This proof consists of two parts: The first part is a AND-combination of L proofs that the i-th component is an encryption of either "0" or "1". The second part is the proof that the sum of all components is equal to 1. These AND- and OR-combinations of proofs can be generated using [**?**].

$$\begin{aligned}\text{proof } \tilde{P} = ((&E(v_{\pi_{\mathrm{voterID}}(i)}, \alpha_{\pi_{\mathrm{voterID}}(i)}) \text{ is encryption of 0 or 1} \mid i = 1 \ldots L),\\ &(\textstyle\bigotimes_{i=1}^{L}(E(v_{\pi_{\mathrm{voterID}}(i)}, \alpha_{\pi_{\mathrm{voterID}}(i)}) \text{ is encryption of 1})))\end{aligned}$$

6. To create his ballot, the voter concatenates his voterID, the voteID, the encrypted and permuted vote $\tilde{e}$, the authentication tag T and the validity proof $\tilde{P}$. He signs the whole string with his secret key $\mathrm{SK}_{\mathrm{voterID}}$. The ballot consists of the five concatenated parts and the signed part.

$$\text{ballot } b = (\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P}, \text{sign}_{\mathrm{SK}_{\mathrm{voterID}}}(\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P}))$$

7. The voter sends the ballot to those authorities he wants to. To be sure that his ballot is counted, he needs to send it to at least one honest authority.

Analysis A voter is allowed to cast several ballots, as long as there are not two or more which are both correct and valid, but different from each other. In this case none of them will be counted. If a voter is coerced to cast a certain ballot, he has the possibility to make sure that it will not be counted by using wrong π_{voterID} and $K_{\mathrm{auth}_{\mathrm{voterID}}}$ as described in the denying phase. Later the voter can still cast another ballot that is correct and valid and will therefore be counted, as long as he can be sure that the coercer will not be able to notice this second ballot.
If the voter misbehaves, this will result in an incorrect (e.g. in case of choosing an illegal vote and therefore creating an incorrect validity proof) or an invalid ballot (e.g. in case of using a wrong π_{voterID} or $K_{\mathrm{auth}_{\mathrm{voterID}}}$. This ballot will not be counted or in case of incorrectness simply be ignored.

3.4.4 Denying Phase

This phase is optional. If the voter wants to sell his vote to a votebuyer, he has to prove that the vote he casted is the one the votebuyer is willing to pay for.
No matter whether the voter has casted this vote or any different one, he is always able to prove that it was the one the votebuyer asks for by using the fake protocol. This will give him a different authentication key $K'_{\mathrm{auth}_{\mathrm{voterID}}}$ that fits perfectly to the ballot he recently sent to the authorities and to the candidate permutation π'_{voterID} the user claims to have received.

As there is no possibility in deciding whether a pair π_{voterID}, $K_{\mathrm{auth}_{\mathrm{voterID}}}$ comes from the authority or from the fake protocol, the proof he gives to the votebuyer is worthless.

1. The voter uses the fake protocol to generate $K'_{\mathrm{auth}_{\mathrm{voterID}}}$ for each votebuyer he wants to, according to the desired vote.

$$K'_{\mathrm{auth}_{\mathrm{voterID}}} = \text{generateFakeKey}((\tilde{e}, \pi_{\mathrm{voterID}}), K_{\mathrm{auth}_{\mathrm{voterID}}}, (\tilde{e}, \pi'_{\mathrm{voterID}}))$$

2. The voter sends the $K'_{\mathrm{auth}_{\mathrm{voterID}}}$, the desired permutation π'_{voterID} and the randomness he has used for the encryption together with his ballot to any votebuyer he wants to convince.

Analysis The voter will be able to convince the votebuyer even if the votebuyer has a copy of the casted ballot, as $K'_{\mathrm{auth}_{\mathrm{voterID}}}$ and π'_{voterID} could have been used to generate the same ballot as well.
Even in case that a coercer is able to interrupt the connection and stop a ballot on his way from the voter to the authorities, he will not be able to force the voter to cast the vote he wants him to. If he asks the voter about his $K_{\mathrm{auth}_{\mathrm{voterID}}}$ and π_{voterID}, the voter will give him the ones from the fake protocol. If the coercer now asks the voter to cast another ballot using the previously stated $K'_{\mathrm{auth}_{\mathrm{voterID}}}$ and π'_{voterID}, the ballot will be invalid, as the authorities will concatenate the encrypted and permuted vote $\tilde{e}$ with the correct π_{voterID} and therefore the authentication tag T will not match.

3.4.5 Collection Phase

During this phase each authority stores all the ballots she receives and which have a correct signature.

1. For each received ballot, each authority checks whether the signature is correct according to the $\mathrm{PK}_{\mathrm{voterID}}$ from the voter who's voterID is included in the ballot. If it is not, this ballot is ignored. Otherwise the ballot is stored.

   If he receives an identical ballot that he has already stored, the new one is ignored.

Analysis This could take a huge amount of storage if many voters would send many ballots with correct signatures in a kind of denial-of-service attack. As this would take place either in case of coercion of many voters, which is not very likely as coercion usually does not work at large scale, or in case of a huge amount of dishonest voters, it is rather not to be expected.
If a signature is correct the ballot must have been created by the voter or by someone who knows his secret key $\mathrm{SK}_{\mathrm{voterID}}$ as a signature cannot be falsified by assumption.
At the end of this protocol we will present an extension with incremental tallying, where the problem of a denial-of-service attack is weakened.

3.4.6 Agreement Phase

Up to now the authorities do not have a common view of the casted ballots. After this phase until the end of protocol, there is a common view between all honest authorities upon the submitted ballots.

1. At the end of the vote, each authority broadcasts the stored ballots to all other authorities over the secure channels between the authorities.

2. All the received ballots are stored.

Analysis If an authority A_i either does not broadcast or does not store a ballot, then $A_i \in \mathcal{A}_{\mathrm{bad}}$ and therefore acts in an arbitrary way and the result of such an authority could be arbitrary anyway. As long as the voter has sent his ballot to at least one honest authority, he can be sure that his vote will be broadcasted and therefore be counted by all honest authorities.

3.4.7 Tallying Phase

In this phase the encrypted result is calculated. The ballots from every voter have to be considered and each authority has to decide which votes are correct and valid and should therefore be counted. To be able to detect the number of irregularities during the vote a list of accusations l is used. l is a vector of length N, one dimension for every authority $A_1 \dots A_N$. This vector is used for storing the information how many voters, corresponding to the authority they are assigned to according to the random function $\mathrm{rf}_{\mathrm{seed}}$ have casted correct ballots but only invalid ones.
Each authority processes the ballots from each voter in the following way:

$$\text{ballot } b = (\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P}, \text{signature})$$

1. All received ballots are checked for correctness. That means that the voterID must be the one of an entitled voter, the signature must be correct corresponding to the voters public key, the voteID has to be the one of the current vote and the validity proof is correct. If a ballot is not correct, the authority will not consider it any further.

$$\text{verify}((\text{voterID}, \text{voteID}, \tilde{e}, T, \tilde{P}), \text{signature}, \text{PK}_{\text{voterID}}) \rightarrow \{\text{true}, \text{false}\}$$

2. All received correct ballots are checked for validity, that is whether verifyTag($K_{\text{auth}_{\text{voterID}}}, (\tilde{e}, \pi_{\text{voterID}}), T$), as defined in Section **??**, returns true.

3. Now there are four possible cases for each voter:

   - If the authority has no correct ballot from a voter, nothing is counted.
   - If she has one or several correct, but invalid votes and no correct and valid ones, she increases the number of accusation for the authority which is assigned to this voter according to the random function rf_{seed} by one in the list of accusations l.
   - If he has exactly one correct and valid ballot, that ballot will be counted. If there are invalid ones, they can be ignored. The encrypted and permuted vote is re-permuted using ${\pi_{\text{voterID}}}^{-1}$ and added homomorphically to the intermediate result.
   - If an authority has several different correct and valid ballots from one voter, nothing is counted.

Analysis At the end of this phase the encrypted tally all honest authorities will calculate is an encryption of the result of at least all those correct and valid votes casted by entitled voters, that reached an honest authority.

3.4.8 Decryption Phase

1. All honest authorities will get the same result that will now be decrypted jointly as defined in Section **??**.

2. The decrypted result has to be published authentically together with the witness to prove that this is the correct decryption.

Analysis An authority A_i, where $A_i \in \mathcal{A}_{\text{bad}}$ will act in an arbitrary way and will get any result she wants to. As long as the invariant $\mid \mathcal{A}_{\text{bad}} \mid < \frac{N}{2}$ holds it will be possible to figure out the correct result among all the published results, as the majority of all honest authorities will get the same one.
If an authority misbehaves in Step 1. during the decryption, there are still at least the $(N - t)$ honest ones that will be able to decrypt the result.

3.5 Security Analysis

3.5.1 Correctness

An honest voter will cast a correct ballot. If he has received the same $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} as A_1 has broadcasted in the initialization phase from his corresponding authority, the ballot will be valid and therefore be counted. If he has wrong ones the ballot will be ignored and the corresponding authority is accused. The misbehavior is detected but correctness is affected.

As it is impossible by assumption on the broadcast protocol that the honest authorities receive different public keys for a certain voter or different $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} from the initiator, they will all consider the same ballots as correct and valid. Therefore they will all get the same encrypted result.

The number of honest authorities is greater or equal the threshold t by assumption. For this reason the honest authorities will be able to decrypt the result even without any help from a dishonest authority, as any number of authorities greater than t can decrypt the tally with their shares z_i.
At the end of the decryption phase all the decrypted results are published authentically. At least every honest authority will now publish her result. As all their results are equal and the number of honest authorities is the majority, whoever knows all published results can figure out which must be the correct result.
If the assumption of the majority of the authorities being honest is violated, the main problem is that the honest authorities are not able anymore to decrypt the encrypted tally as for the decryption protocol, as described in Section **??**, at least t participating authorities are needed. And even if some dishonest authorities joined in the decryption, the interpretation of the result which is published at the very end would be arbitrary. If all dishonest authorities decide on the same result that might have been specified even before the vote took place, an observer will not be able to distinguish whether the majority consists of the honest ones and some of the dishonest ones, or if the correct result is published only by a minority.

3.5.2 Privacy

The secrecy of an encrypted vote is guaranteed as long as the assumption holds that $\mid \mathcal{A}_{\text{bad}} \mid < t$. Any set of less than t authorities will not be able to decrypt a vote.
The only information that everybody can get who is able to eavesdrop all the channels between voters and authorities is a kind of traffic analysis to see which voter casted which number of ballots and whether they were correctly signed.

The number of ballots the voter sends and—for the dishonest authorities who know $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} —the fact whether a ballot is valid or not could be analyzed. As this could have different reasons, from a simple mistake from a voter while generating his ballot to a voter being coerced, the gain of information is limited.
If the assumption that the number of dishonest authorities is bound by the threshold t is violated, privacy can no longer be guaranteed. The dishonest authorities are now able to decrypt every vote and as the candidate permutation π_{voterID} is known to all authorities to determine the candidate the voter casted his vote for.

3.5.3 Receipt-freeness

A voter is able to "prove" to a votebuyer, that he casted exactly that vote the votebuyer asks for, even if a votebuyer is able to eavesdrop all casted ballots on all channels between voters and authorities and the voter has casted a different vote. Using the fake algorithm as described in Section **??** he can create a string that fits with the vote he claims to have casted perfectly.

If a voter is forced to use a certain $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID} to create his ballot or even to cast a ballot a coercer already created, he is able to make the ballot invalid or in the second case he is already invalid with overwhelming probability. His ballot is ignored as if he would not have voted at all, except for the accusation that is a kind of indicator that there could have been something going wrong. It is impossible for a coercer to force a voter to cast a ballot that will influence the result. The strongest attack a coercer can do is to prevent a voter from voting, what he could do in the classical setting too.

The assumption that the sheet with the $K_{\text{auth}_{\text{voterID}}}$ and the π_{voterID} will not be exchanged on his way from the authority to the voter by mail is a realistic as it would not make much sense to change

these informations only for a single voter. If it is done for a huge number this will be detected by the number of accusations and maybe a vote would even have to be repeated.
Given the number of dishonest authorities is violated, it has no effect on the receipt-freeness. If the assumption that no authority may cooperate with a votebuyer is violated, the protocol loses its receipt-freeness. Even if only a single authority cooperates with a votebuyer and even if it is known which one (if that one is not excluded from the set of authorities), receipt-freeness is no longer guaranteed, as every authority knows the secret information that is needed for the receipt-freeness, $K_{\text{auth}_{\text{voterID}}}$ and π_{voterID}. A votebuyer who gets this information can no longer be convinced with $K'_{\text{auth}_{\text{voterID}}}$ and π'_{voterID} from the fake protocol.

3.6 Incremental Version of the Protocol

This is an improvement of the basic version of the protocol. The main advantage lies in the efficiency. The initialization phase, the distribution phase, the voting phase, the denying phase and the decryption phase remain exactly as they were in the basic version.

3.6.1 Modifications compared to the Basic Version

The collection phase, the agreement phase and the tallying phase are merged into one single phase. The advantage is — except for performance reasons — that all honest authorities have a common view at any point of this phase and the intermediate result, the encrypted sum, is always correct according to the received ballots.
To process the ballots in this way, a kind of synchronicity is needed that is stronger than the one in the basic version. In the basic version the only point that has to be reached synchronously is the beginning and the end of each phase. In the incremental version each broadcast from one authority to all the others has to follow a synchronous model.
The definitions of a correct and a valid vote remain the same as defined in the basic version.

1. Each authority stores for each voter one of four possible states, beginning with the initial state empty.

   - state empty: No correct ballot from this voter has yet been received. This is the initial state.[7]
   - state invalid: Up to now, the authority has received one or several correct but invalid ballots, but none that was correct and valid.
   - state valid: The authority has received exactly one correct and valid ballot (maybe beside other incorrect or correct but invalid ballots)
   - state double: There have been two or more different ballots that are correct and valid as well.

2. Every time a new ballot is received, the authority processes it in the following way:

   (a) The ballot is checked for correctness.
   (b) The ballot is checked for validity.
   (c) The authority acts according to the state the voter is in.

   If a ballot is incorrect, it is ignored and will therefore have no influence on the actual state. If the ballot is correct it is processed in the following way:

[7] If an incorrect ballot is received one does not know at all if it has been created by the voter who's ID is included in the ballot.

	correct, but invalid ballot received	correct and valid ballot received
state empty:	ballot is stored, new state: invalid	ballot is stored, $\tilde{e}$ is permuted and added homomorphically to the intermediate result, new state: valid.
state invalid:	ballot is ignored, remains in state invalid	ballot is stored, instead of the previous one, $\tilde{e}$ is permuted and added homomorphically to the intermediate result, new state: valid.
state valid:	ballot is ignored, remains in state valid	if the ballot is identical to the one already stored, it is ignored and remains in state valid. If the ballot is different from the stored one, it is stored additional to the other, the previously stored one is permuted and subtracted homomorphically from the intermediate result, new state: double.
state double:	state double is a trap state. No matter what is received, everything is ignored and it remains in state double	

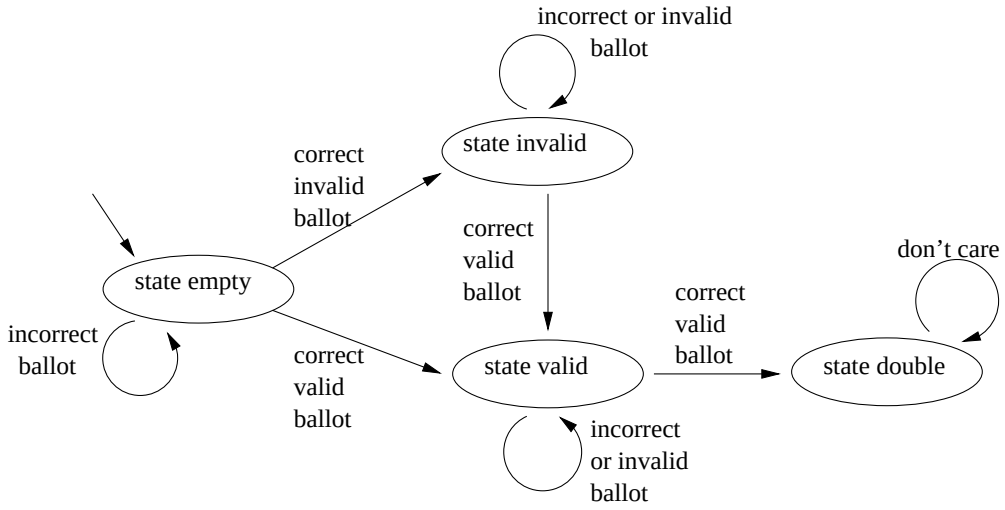


Figure 3: State diagram for incremental collection

3. Each time a change of state is processed, the authority broadcasts the received ballot over the secret channels to all other authorities.
   An authority who receives a ballot broadcasted from another authority processes it exactly in the same way as described in Step 2.

   After this step all honest authorities have the same view on all submitted ballots.

4. For every voter in state invalid the corresponding authority is accused.

3.6.2 Analysis of the Incremental Version

The property of correctness will stay in the incremental version exactly as it is in the basic version. Each vote that is counted in the tally would have been counted in the basic version too. The ballots that are counted in the basic version are the ones that are correct and valid and where no second

ballot from the same voter that is different from the first one exists that is correct and valid too. In the incremental version this means that at some point in time the transition from either state empty or state invalid to state valid is taken. The only reason for making this transition is that a correct and valid ballot has arrived. The only transition that leads away from state valid leads to the state double. As there is no second ballot with the same properties but different from the first one, the transition will not be taken and therefore the vote is counted in the incremental version too.
Every vote counted in the basic version will influence the result in the incremental version in the same way. If no ballot from a certain voter is counted it means that either no valid ballot has been received or two or more different correct and valid ballots have been casted by the same voter. In the first case, that voter will not reach the state valid and therefore no vote from him will be counted. In the second case all those ballots will arrive at all honest authorities too and the voter will remain in state double for the rest of the vote.

Neither privacy nor receipt-freeness are influenced by the changes from the basic to the incremental version as no entity gets more information than it would have gotten in the basic version.

4 Summary

The protocol that we propose seems to achieve less than other protocols due to the weakening of the correctness property only to detect misbehavior instead of preventing it. It does not provide universal verifiability. Yet another disadvantage is the strong restriction that no authority may cooperate with a votebuyer or coercer.

Taking a closer look at other protocols claiming to achieve these goals we determine that — at the point where one implements and uses these protocols — they lose many of their properties as they have to compromise with the real world.

In order to get universal verifiability one needs an additional device where all votes are collected, such as a bulletin board. As there is no such thing it has to be simulated by someone. An obvious solution would be to do this by the set of authorities. But at this point we have lost our property of universal verifiability and we are in the context of a threshold scheme as it is used in our protocol because in case that the number of dishonest authorities reaches a certain level, the bulletin board could be manipulated.
In order to achieve receipt-freeness without the restriction of the cooperation of authorities and votebuyers, strong physical assumptions are needed. But these things do not (or not yet?) exist in the digital world or are not available at large scale. To realize an untappable channel for each entitled voter means to force him to have physical contact e.g. by making a visit at the local government. A voting booth does exist, but only in the physical world and there is no implementation for it for an electronic context. This is a step back that would not be approved by the voters. To convince a voter to cast his ballot using the electronic way there have to at least no disadvantages compared to the status quo.

The only restriction why electronic voting as we propose it cannot take place in near future is the assumption of a public-key infrastructure. All other preconditions are fulfilled. This is the strength of our protocol.

It seems to be the weakest point in the protocol that a dishonest authority can prevent a voter from casting a vote. It would not make much sense for an authority to do this to a single voter. This would cause a single accusation and would therefore be interpreted as some misbehavior of a

voter or a case of coercion rather than a misbehavior of an authority, but there would be only a very small change in the result. If an authority prevents a whole group from casting their votes, it is likely that this would be noticed by looking at the list of accusations and could have some effect for this authority.

The problem how to handle the cases where non-negligible numbers of accusations either against one authority or against several or even all authorities is difficult. A political solution would be needed which is accepted by the community. One way would be to compare the total number of accusations with the differences between the results for the different candidates. If the total number of accusation is small enough not to change anything in the result, it could be considered correct. But what about votes where only a very small majority decides which way to go and the number of accusations reaches a level that could have influenced the result? Should the vote be repeated?

Which are the advantages of the protocol compared to votes as they take place today e.g. in Switzerland? The advantage that it would be more comfortable to cast a vote over the internet is often mentioned, but it seems to be rather small as it would surely not be much easier than casting the ballot by mail which is possible in classical votes today. A further advantage might be that voting over the internet could take place at the very time the vote takes place and does not have to be done several days before. In questions of trust both possibilities seem to be similar. If a voter does not trust any authority he will not be sure that his vote will be counted in either of the two ways. The only solution for this would be universal verifiability which cannot be realized. An advantage worth mentioning is that particularly in the incremental version the tallying is finished more or less at the same moment when the vote is closed.

The property of receipt-freeness should not be underestimated even in comparison with the conventional vote. With the possibility to vote by mail votebuying is much more likely than it was before. A votebuyer could pay everybody who sends him all the documents for the vote together with his document of identification for this vote that the voter has to sign beforehand. The votebuyer would check the ballot and send it to the government himself and pay the voter if everything was fine. Regarding this attack our protocol would be quite an improvement although all kind of votebuying in the physical world does not seem to work in large scale.

Compared to the most simple model with a central server that collects and decrypts all votes and presents a tally at the end of the vote, our protocol does not need such assumptions of trust from the voter's side. If that central server "acts" dishonestly in any way the result is arbitrary but nobody can detect any irregularity.

From a voter's point of view the procedure is very similar to a conventional non-electronic voting as it currently takes place e.g. in Switzerland. It is based only on technical assumptions that are fulfilled today except for the public-key infrastructure that is needed to guarantee the authenticity of the different messages.

As a conclusion our protocol, is a real alternative to the system of voting as it is used nowadays. It could be realized without huge effort and simply by using the electronic infrastructure that is at hand in contrast to other protocols achieving receipt-freeness.

References

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104, 1997.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology — EUROCRYPT '97*, Lecture Notes in Computer Science, 1997.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, 1984.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer Security*, 28:270–299, 1984.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, September 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *Proc. of IFIP '96, Advanced IT Tools*, pages 21–30. Chapman & Hall, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. of Workshop on Security Protocols '97*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35, 1997.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology — EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, 1999.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract). In *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, 1991.

[RSA] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

A Project Description

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute for Theoretical Computer Science
Prof. Ueli Maurer
Martin Hirt

Semester Project for Thomas Briner
March 31, 2003 – July 4, 2003

Electronic Voting in the Standard Model

1 Introduction

In a voting scheme, every entitled voter can submit one vote, and the sum of the submitted vote is computed and published. A secure voting scheme should satisfy (at least) the following properties:

- SECRECY. It is infeasible to find out which voter has submitted which vote.
- CORRECTNESS. The published tally is the correct sum of the submitted valid votes.
- VERIFIABILITY. The correctness of the published tally can be verified.

The correctness property includes *eligibility* of the voter and *validity* of the vote. Verifiability can be either *local*, i.e., every voter can verify that his ballot is counted, or *universal*, i.e., anyone can verify that all submitted ballots are counted. Secrecy may include *anonymity*, meaning that one cannot determine whether or not a certain voter has participated the vote, and may also include *receipt-freeness*, meaning that a voter cannot construct a proof of the submitted vote. The latter is important for preventing vote-selling, as it disables the voter from convincing a vote-buyer that the submitted vote is as required.
In a voting protocol, the above properties are satisfied *under assumption*: Typically, voting protocols involve several authorities, and correctness and secrecy are satisfied under the assumption that a certain number of these authorities are honest. Verifiability is usually guaranteed under the assumption of the availability of a *bulletin-board*, a broadcast channel with memory. Receipt-freeness is often not achieved. If it is satisfied, then usually under some (rather unrealistic) assumptions on the underlying communication model, e.g., existence of a voting booth or of untappable channels. Known protocols in the literature do not satisfy anonymity.
Electronic voting was first proposed by Chaum [Cha81]. Subsequently, many voting protocols were published. The concept of receipt-freeness was introduced by Benaloh and Tuinstra [BT94]. Receipt-free protocols were published in [SK95, Oka97, HS00, Hir01, BFP$^+$01].

2 Description

Voting protocols in the literature provide receipt-freeness (if at all) only under some unrealistic assumptions. Actually, full-strenght receipt-freeness seems unachievable in the standard model. However, many receipt-free protocols even fail completely (also with respect to secrecy and/or correctness) when the additional asusmptions are not met; hence provide a lower security level in a real-world environment than classical voting protocols.
The goal of this work is to construct a voting scheme which is “as receipt-free as possible” under realistic assumptions. Furthermore, the new scheme should not have any disadvanatage compared to classical voting schemes. Ideally, one can constructed a protocol with “staggered receipt-freeness”. Under standard assumptions, the standard level of security (but no receipt-freeness) is achieved. Some limited level of receipt-freeness is achieved under some reasonable assumption, and the full level of receipt-freeness is achieved under some (perhaps unrealistic) strong assumption.
As starting points, the generic framework for constructing homomorphic (receipt-free) voting protocols [Hir01] should be considered. Furthermore, the work on deniable encryption [CDNO97] and uncoercible multi-party computation [CG96] might be helpful for reducing the assumptions.

3 Tasks

The following is an (incomplete) list of tasks:

- Study of the literature, in particular [BT94, SK95, CDNO97, CG96, HS00, Hir01, BFP+01],
- construct voting protocols under standard assumptions with highest possible security level,
- analyze and prove the security of the proposed protocols.

The results have to be presented in a talk by the end of the project. Some instructions about the documentation can be found in the enclosed leaflet.

References

[BFP+01] Olivier Baudron, Pierre-Allain Fouque, David Pointcheval, Guillaume Poupard, and Jacques Stern. Practical multi-candidate election system. In *Proc. 20th ACM Symposium on Principles of Distributed Computing (PODC)*, 2001.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th ACM Symposium on the Theory of Computing (STOC)*, pages 544–553. ACM, 1994.

[CDNO97] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption. In *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 90–104, 1997.

[CG96] Ran Canetti and Rosario Gennaro. Incoercible multiparty computation. In *Proc. 37th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 504–513, 1996.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting*. PhD thesis, ETH Zurich, 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology — EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556, 2000.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proc. of Workshop on Security Protocols '97*, volume 1361 of *Lecture Notes in Computer Science*, pages 25–35, 1997.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme – A practical solution to the implementation of a voting booth. In *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403. Springer-Verlag, 1995.