Department of Computer Science

Faculty of Mathematics, Physics and Informatics

Comenius University, Bratislava

Zuzana Rjašková

Electronic Voting Schemes

Diplomová práca

April 2002

Abstract

Electronic voting is an application of cryptography. Voting schemes that provide receipt-freeness prevents voters from proving their cast vote, and hence thwart vote-buying and coercion. We revise the contemporary state of research in electronic voting and propose an efficient receipt-free voting scheme. Similar to the scheme of Hirt and Sako, it assumes the existence of untappable communication channels between the voter and the authorities. Compared to the receipt-free scheme of Hirt and Sako, the scheme described in this paper realizes an improvement of the total number of bits sent through the untappable channel by a factor L (number of possible votes/choices) while achieving the same security properties. We also discuss an implementation of the untappable channel.

Contents

1 Introduction

Through the centuries, different voting technologies have done their best. Stones and pot shards dropped in Greek vases led to paper ballots dropped in sealed boxes. Nowadays, new technologies are developed to automate the voting process. The automation should preserve the security of the traditional elections (especially the privacy of the votes). Mechanical voting booths and punch cards are already designed to replace paper ballots for faster counting.

Electronic online voting over the Internet would be much more profitable. Many voters would appreciate the possibility of voting from anywhere. Convenience of the voting will result in increasing the number of participating voters. Fast, cheap and convenient voting process could have great impact on the contemporary democratic societies. For instance, elections could be held more often in order to allow the citizens to express their will at any time.

Electronic voting has been intensively studied for over the last twenty years. Up to now, many electronic voting schemes have been proposed, and both the security as well as the effectiveness have been improved. However, no complete solution has been found in either theoretical nor practical domains.

The aim of our work has been to revise the contemporary state of research in the field of electronic voting, and to introduce our own solution enhancing effectiveness.

The most efficient voting protocols could be categorized by their approaches into two main types: schemes using blind signatures and schemes using homomorphic encryption. The suitability of each of these types varies with the conditions under which it is to be applied.

In the schemes using blind signatures, the voter firstly obtains a token – a blindly signed message unknown to anyone except himself. Next, the voter sends his token together with his vote anonymously. These schemes require voter's participation in more rounds.

In the schemes using homomorphic encryption the voter cooperates with the authorities in order to construct an encryption of his vote. Due to the homomorphic property, an encryption of the sum of the votes is obtained by multiplying the encrypted votes of all voters. Finally, the result of the election is computed from the sum of the votes which is jointly decrypted by the authorities.

A voting scheme must ensure not only that the voter *can* keep his vote private, but also that he *must* keep it private. In other words, the voter should not be able to prove to the third party that he has cast a particular vote. He must not be able to construct a receipt proving the content of his vote. This property is referred to as *receipt-freeness*.

Only a few schemes guaranteeing receipt-freeness have been proposed. Known receipt-free scheme using blind signatures [Oka97] assumes the existence of a special anonymous untappable channel. Achieving the communication that is both secure and anonymous would, however, be extremely difficult. As for the schemes using homomorphic encryption, some efficient receipt-free schemes have already been proposed. The most practicable one seems to be that of Hirt

and Sako from [HS00].

The goal of our work was to design a scheme more effective than the above mentioned scheme of Hirt and Sako and achieving the same or higher security properties.

The problem of electronic voting is formulated in the section 2. The next section 3 briefly introduces basic approaches to the problem. Necessary cryptographic primitives and interactive proofs can be found in the section 4. Overview of the existing voting schemes, as well as a description of the chosen schemes can be found in the section 5. Our proposed scheme is based on the work of Hirt and Sako (section 5.4) and is discussed in the last section 7. The concept of deniable encryption and the implementation of the untappable channel can be found in the section 6.

2 Formulation of the problem

2.1 Traditional Elections

Traditional elections have some important features, which the electronic voting, in order to be useable, should have also. We briefly sketch the most important ones of them.

Voting committee takes care of voters: It allows only eligible voters to vote, and it ensures that every voter votes at most once. After the elections, voting committee counts the votes and publishes the result. The votes remain secret – no one is able to say how John Smith has voted (We know that he has voted – we have seen him, and his name in the list of eligible voters is marked.) Of course, when ninety–nine percent of voters say no, everybody knows that John Smith said with high probability no. We simply can do nothing with the deduction from the result of election. Even if John Smith tells his vote to Mary Carpenter, she will not believe him – he can easily lie. On the other hand, John Smith cannot be absolutely sure that his vote was really counted. He can just believe it was. Everybody has to believe that the voting committee is honest and it would not disrupt the elections.

2.2 Basic Model

Participants in our schemes will be voters and authorities. We see both of them as probabilistic Turing machines, which can perform probabilistic polynomial-time computation. Participants can communicate with each other through the public channel. We will denote the number of voters M and the number of authorities N. M can be much higher than N.

2.2.1 Voters

In general, voters are not willing to bother with complicated and time-spending voting process. Therefore voter's actions and computations during the electronic voting should be kept at minimum, realizing vote-and-go concept. Voter can abstain from voting if he wishes to – he need not participate in the voting, or he can stop his voting any time before it is finished (of course, in this case his vote is not counted). Further, we suppose that he can secretly store some amount of data in a secure place inaccessible to anyone except for himself.

2.2.2 Authorities

Authorities manage the elections. They have large computing power and they can store large amount of data in secret. Authorities can also act as voters. Maximum number of the faulty authorities will be denoted as t. We assume that the rest $N - t$ from N authorities will do their prescribed work correctly and honestly.

2.2.3 Votes

The structure of votes depends on the type of elections. More precisely, it depends on the question that is put forward to voters in the election and possible answers.

We will distinguish between the following types of election:

- *yes/no voting.* Voter's answer is yes or no. Vote is a one bit: 1 for yes and 0 for no.
- 1-*out-of-L voting.* Here, voter has L possibilities and he chooses one of them. Vote is a number in the range $1 \dots L$.
- K-*out-of-L voting.* Voter selects K different elements from the set of L possibilities. The order of the selected elements is not important. Vote is a K-tuple $(v_1 \cdots v_K)$.
- K-*out-of-L ordered voting.* Voter puts into order K different elements from the set of L possibilities. Vote is an ordered K-tuple $(v_1 \cdots v_K)$.
- 1-L-K *voting.* Voter picks out one of the L sets of possibilities, and from the selected set he chooses K elements. Vote is a $K+1$-tuple $(i, a_1 \cdots a_K)$; $a_1 \cdots a_K$ are elements of the ith set.
- *Structured voting.* There are n levels of possibilities. Voter moves from the first level to the last one. At the ith level he can select at most k_i possibilities from the subset S_i of all possibilities in the ith level. S_i, k_i depend on his choices in the previous levels. Vote is a tuple $(v_{11}, \cdots, v_{1k_1}, \cdots, v_{i1}, \cdots, v_{ik_i}, \cdots, v_{nk_n})$, where $\{v_{i1}, \cdots, v_{ik_i}\} \subset S_i$.
- *Write-in voting.* Voter formulates his own answer and writes it down. Vote is a string with specified maximum length.

For example, possible reply to the question "Do you agree with ..." or "Do you wish that ..." can be only "Yes" or "No" (yes/no voting). An example of 1-out-of-L voting is choosing a leader from a list of L candidates (e.g. presidential elections).

Further, imagine the election of council members, in which the voter selects K from L candidates. Those candidates that were selected the most times will become council members. In this case, the order of the voter's K selected candidates is not important and therefore this type of voting belongs to the K-out-of-L voting. Another way how the votes in this election could be evaluated is to take into account the order of the voter's candidates: the candidate who is marked by the voter as first will get the most points, the candidate marked as the last (Kth) the least points. Those candidates with the largest numbers of points will become the members of the council. This type of election is a typical example of K-out-of-L ordered voting.

Somewhat special seems to be parliamentary election, when the voter elects his representatives, but they have to be candidates of the same political party. This is precisely 1-L-K-voting, where L is a number of political parties and K a number of the selected candidates. Again, the selected set may be ordered or unordered.

Structured voting is a generalization of 1-L-K voting, and can be seen as filling in the form, where the possibilities for the next part depends on the previous choices.

Notice that 1-L-K voting is structured voting with two levels: first contains L possibilities, $k_1 = 1$ and $k_2 = K$. In fact, voter's choice in all types of voting is one element from the finite set of possibilities. For example, in K-out-of-L voting voter selects one K-tuple from the set of all K-tuples. Therefore, every type of voting can be seen as 1-out-of-L voting for some L. However, L could rise too high and this approach can turn to be impractical.

Moreover, according to the equality of the votes we define two types of voting:

- *equal-voting.* Each voter can vote only once and his vote is counted once.
- *weighted-voting.* The vote of the voter V_i is counted w_i times.

Notice that the weighted-voting cannot be realized simply by repeating voting procedure w_i times, as it gives the voter the opportunity to vote every time differently.

In the equal-voting the power is distributed equally among the voters, for instance in parliamentary elections. In weighted-voting (e.g. at the general meeting of the stockholders) the power to make decision is not equally distributed, and the voters are more or less privileged.

A structure containing the vote is called a ballot. It can be easy, difficult or impossible to extract the vote from the ballot, depending on the scheme.

2.2.4 Trust

If we have one absolutely reliable authority, about which we are sure it will make the expected actions and it will do nothing else (e.g. release some secret information, cast the ballots of voters abstaining from voting, or abuse its role in any other manner), we do not need more authorities. But the situation is different in the real world. We find it risky to put the whole responsibility for the elections to the one authority. Therefore we share the ability to lead the elections between more authorities in such a way that malicious behavior of some of them will not jeopardize the elections.

Each participant (voter as well as authority) has to believe that at least $N - t$ authorities are honest. This trust can be a sort of "general suspicion", when (from the participant's point of view) every authority is dishonest with the same probability, but the participant believes that the actual number of the dishonest authorities will not exceed t. Authorities do not trust voters at all.

2.2.5 Communication

In this section, various types of communication channels used in electronic voting schemes are introduced.

Any participant can send a message to any other participant through the public channel. *Bulletin board* is publicly readable. Any participant can write in (only in his own section), but nobody can delete or change anything in the bulletin board. Bulletin board can be considered as public channels with memory.

Untappable channel is a secret channel between two participants. Communication through untappable channel is physically secure: no one else can see or change the sent message, and even the participant cannot later demonstrate to anyone what was sent. The existence of the untappable channel is assumed in some schemes between the voter and the authority. For implementation of the untappable channel see section 6.

Untraceable anonymous channel , or *anonymous channel* for short, is a channel guaranteeing the anonymity of the sender. Recipient of the message that has been sent through the anonymous channel does not know the identity of the sender. No one is able to trace the message back to the sender. Realization of this channel will be discussed later in section 4.11. Note that the anonymous channel needs not to be untappable (e.g. the messages arriving to the receiver can be tapped).

Untappable anonymous channel is a channel guaranteeing both the anonymity of the sender and the physical security of the transmission: the sender / receiver cannot later demonstrate what was sent / received. No one can intercept the transmission of the message. Implementation of the untappable anonymous channel is hard in practice.

2.3 Electronic Voting Scheme

The authorities and the voters have to follow electronic voting scheme. The scheme prescribes voter's and authority's actions and computations during the voting process. We assume that at least $N - t$ from N authorities will not deflect from the expected behavior. The scheme should be resistant to malicious actions of the other t authorities. Voters can act on their own will. The aim is to design the scheme in such a way that malicious or improper behavior of the voter will be detected, and invalid or double-votes will not be taken into account.

2.4 Definition

Electronic voting scheme consists of three main stages: initialization stage, voting stage, and counting stage. The stage can consists of more *phases.*

Initialization stage. At this stage, authorities set up the system. They announce the elections, formulate the question and possibilities for an answer, create a list of eligible voters, and so on. They generate their public and secret keys, and publish the public values.

Voting stage. Voters are casting their votes. The voter communicates with authorities through the channels he can use, forming a ballot containing his vote. Finally he sends his ballot to its destination.

Counting stage. Authorities use their public and secret information to open the ballots and count the votes. They publish the result of elections.

2.5 Requirements

In order to be usable in practice, electronic voting scheme has to satisfy some requirements.

Eligibility. Only eligible voters can cast the votes. Every voter can cast only one vote.

Privacy. No coalition of participants (of reasonable composition) not containing voter himself can gain any information about the voter's vote. By reasonable composition we mean coalition of at most t authorities and any number of voters. We say that information-theoretic privacy is achieved when the ballots of the voters are indistinguishable independent of any cryptographic assumption; otherwise we say that computational privacy is achieved.

Individual verifiability. Each eligible voter can verify that his vote was really counted.

Universal verifiability. Any participant or passive observer can check that the election is fair: the published final tally is really the sum of the votes.

Fairness. No participant can gain any knowledge about the (partial) tally before the counting stage (the knowledge of the partial tally could affect the intentions of the voters who has not yet voted).

Robustness. Faulty behavior of any reasonably sized coalition of participants can be tolerated. No coalition of voters can disrupt the election and any cheating voter will be detected.

Receipt-freeness, incoercibility. We say that the scheme is incoercible if the voter cannot convince any observer how he has voted. This requirement prevents vote-buying and coercion. Before the election, someone can bribe or coerce the voter to vote in a particular way. The coercer can order the voter how he should behave during the voting process (e.g. generates for him random bits). During the election, the coercer can observe the public communication between the voter and the authorities. After the election, he will want to see a proof that the voter really voted this way. In the scheme achieving privacy, the coercer alone or with reasonable coalition of participants cannot open the voter's vote. Thus the coercer will force the voter to show him his secret information. With it, he is capable of opening the ballot and seeing the vote. Incoercible scheme provides the voter with the ability to modify his secrets and to open his ballot in any desired way. Thus the voter can vote on his own will and he can feed the coercer with a false proof.

3 Various Approaches

We give here an informal overview of the main ideas commonly used in the voting schemes. The sketched approaches do not deal with all possible problems and occasions that sometimes occur. They are just ideas the voting schemes are based upon, and some schemes might use them in various ways.

The most "difficult" property of the voting scheme seems to be privacy. If the requirement of the privacy is omitted, it turns out not to be hard to design a voting scheme that achieves the remaining properties (eligibility, verifiability). (Observe that without the privacy we cannot talk about the receipt-freeness). Just consider the voting scheme in which each eligible voter writes his vote into his own section on the bulletin board, and anyone can read it and count the votes.

Up to now, only a few approaches to achieve privacy have been invented. Privacy means that the link between the voter and his vote is disposed or inaccessible to everyone (including authority), even if all of the public communication is monitored. This can be accomplished in three ways:

- It is easy to see the vote, but it is impossible to trace it back to the voter.
- It is impossible or computationally infeasible to see the actual vote, but it is easy to see the identity of the voter.
- Both seeing the actual vote and obtaining the identity of the voter is impossible or computationally infeasible.

Schemes of the first and the third type have to use some special kind of channel for casting the votes (usually untraceable anonymous channel). In the first approach, the actual votes are published and anybody can count them, but nobody knows who sent which vote. However, a special care is required to achieve eligibility, to ensure that the voter cannot cast more votes and to prevent improper voters from voting. On the other hand, in the schemes of the second type there is a problem of the votes counting – the eligibility is for free. Besides, anybody can see which voters have voted and which have not.

3.1 Schemes Based on Anonymous Channel

Voting stage is a composition of the registration phase and the voting phase. Roughly speaking, the voter obtains a token in the registration phase, which gives him the right to vote in the voting phase.

The voter is not able to create the token by himself, only during the interactivity with the authority. The authority helps the eligible voter to construct the token only once, so the voter could obtain at most one token. The authority has no idea how the voter's token looks like. Moreover, the validity of the token is verifiable to anyone. This concept is realized via blind signatures (described in the section 4.7).

In the voting phase, the voter sends a ballot containing the token and his vote through the anonymous channel to the authority. The authority will not

accept the ballot with invalid token or with the token that has already been used. This ensures that only eligible voters can vote (only eligible voters has been allowed to construct a token), and that they can vote at most once (they can obtain at most one token). As no one (even the authority) can make any connection between the voter and the token or trace the casted ballot back to the voter, no one can deduce anything about how the voter voted (of course, except for the unavoidable deduction from the result of the election). Hence, the privacy is achieved.

In general, token may consist of whatever you want: hidden voter's identity (a hash value for instance), random numbers, encrypted vote, voting tag (unique for each election), etc. The only restriction is that it should be hard or impossible to extract the voter's identity from the token and that each voter has to have different token. Structure of the token is specified in the voting scheme.

The token is sometimes called a pseudonym. The term "pseudonym" emphasizes the fact that the voter's identity and the token (pseudonym) cannot be linked.

This approach can be found in the schemes of [Cha88, PIK93, Boy, FOO92, Rad95, JL97, JLY98, Oka97].

3.2 Schemes Based on Homomorphic Encryption

In this kind of schemes, the voter sends encrypted vote through the public channel (usually to the bulletin board). The vote can be decrypted by any set of at least $t + 1$ authorities, and any set of t authorities can tell nothing about the vote. This can be accomplished in two ways:

- threshold public-key cryptosystem is used for encrypting the votes (A key to decrypt the vote is shared between any set of $t + 1$ authorities, for instance ElGamal cryptosystem, see section 4.10)

- Each authority has its own instance of the cryptosystem. The voter shares the secret (his vote) among the N authorities using $(t+1, N)$ secret sharing scheme (for instance Shamir's scheme from the section 4.2). The voter sends to the each authority its encrypted share.

This will prevent small coalition of malicious authorities to abuse their role and to violate voter's privacy. Problem arises only in counting the votes.

Encryption method used for encrypting votes is homomorphic: multiplication of the encrypted votes is an encrypted sum of the votes (for more details on homomorphic encryption see section 4.9).

In the first case, encrypted votes are multiplied and authorities decrypt only the sum of the votes. In the second case, each authority multiplies its encrypted shares, decrypts the sum of its shares, and the final sum of the votes can be computed by anyone from the $t + 1$ partial sums.

In a yes/no voting, where 1 expresses yes-vote and 0 expresses no-vote, the sum of the votes is the number of yes-votes. As the whole number of votes is known, the number of no-votes is easily computed. For other types of voting we have to think of some more sophisticated encoding of the possible

votes. For instance in an 1-out-of-L voting we can encode the ith possibility ($1 \leq i \leq L$) as M^{i-1} (M is the number of eligible voters). Sum of the votes will be $S_1 = a_1 + a_2 M + a_3 M^2 + \cdots + a_L M^{L-1}$, where a_i is the number of times the ith possibility was sent. Coefficients a_i can be iteratively computed as $a_i = S_i \bmod M$, $S_{i+1} = (S_i - a_i)/M$, where S_1 is the sum of votes ($a_i \leq M$).

The authorities should be able to distinguish between the valid and the invalid encrypted votes (for example, to count his vote twice, the voter can encrypt 2 instead of 1). Invalid votes should be rejected. Usually, the voter is required to prove that his vote is of the correct form (either 1 or 0) without disclosing any other information about his vote (the proof should be zero knowledge).

Usually, schemes based on homomorphic encryption are not receipt-free: let the voter's favorite vote be V, and encryption of the V be C. Voter sends C to the bulletin board, so C is publicly known. Further, suppose that the coercer's favorite vote is W, and that $W \neq V$. Coercer can later force the voter to reveal how the W is encrypted to C. Of course, C is the encryption of V, and the only chance for the voter is to show the coercer that C looks like the encryption of W. This is possible only if the deniable encryption is used. However, a deniable encryption with homomorphic property and suitable to threshold cryptographic techniques is not yet known. This topic will be discussed further in the section 6.

Homomorphic property of encryption method is exploited in these schemes: [Ben87, BY86, Ive91, SK94, CGS97, Sch99, HS00, LK00, FPS00].

3.3 Schemes Based on Mixing the Votes

To be simple, consider 1-out-of-L voting. The authorities take the list of the L possible votes (original list), and mix it to produce the final list. The operation of mixing is performed as follows (see also figure 1):

The first authority takes the list of L possible votes (original list), permutes it in a random order, and re-encrypts each possible vote. It unveils the permutation only to the voter and no one else. To increase the security, the authority sends the permutation to the voter through untappable channel. The created list containing re-encrypted and permuted possible votes is published and handled to the next authority. Seeing just the original and created list, no one is able to say anything about the permutation mapping each item from the original list to its re-encryption in the created list, unless this permutation is revealed to him by the authority.

The next authority takes the handled list, and shuffles it in the same way as the first authority shuffled the original list: it permutes the list in a random order, re-encrypts each item, unveils the permutation to the voter through the untappable channel and publishes the produced list.

Successively, each authority takes the list handled by the previous authority, shuffles it in the manner described above, and handles the produced list to the next authority. The list produced by the last authority is called the final list. Only the voter can keep track of the permutations that have been sequentially

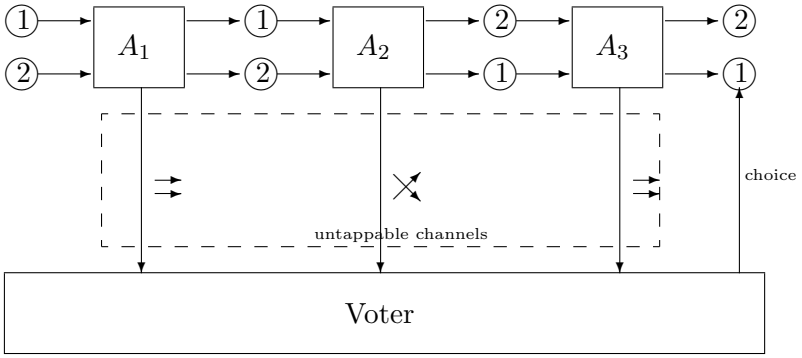

Figure 1: Mixing the votes

applied to the original list by the authorities. Therefore, only he knows the permutation mapping each item from the original list to its re-encryption in the final list. The voter just selects one item from the final list as his vote.

In the case that the used encryption is homomorphic, the voter writes his selected item to the bulletin board. The votes are counted as in the schemes based on homomorphic property: all encrypted votes are multiplied, and the authorities cooperate to decrypt the sum of the votes.

The coercer cannot tap on the secure channel, so the voter can adjust the permutations he received from the authorities as it comes useful. Therefore, the voter is not able to prove how he has voted and these schemes are receipt-free.

Mixing the possible votes and sending done permutations secretly to the voter appear in the schemes of [SK95, HS00].

4 Cryptographic Primitives

4.1 Notation

N	number of authorities
$A_1, A_2, \ldots, A_N$	N authorities
t	maximum number of malicious and dishonest authorities
A	any set of $t+1$ authorities
M	number of eligible voters
m	number of voters participating in the voting; $m \leq M$
$V_1, V_2, \ldots, V_M$	M voters
$v_1, v_2, \ldots, v_M$	intentions (votes) of the voters
Z_p	field of positive integers modulo p, where p is prime number
Z_n	set of integers modulo n, i.e. $\{0, 1, \ldots, n-1\}$
Z_n^*	set of integers from Z_n relatively prime to n
$a \vert b$	an integer a is a divisor of an integer b
$gcd(a, b)$	greatest common divisor of the integers a, b
$a \Vert b$	concatenation of the strings a, b
$a \oplus b$	bitwise exclusive or
$x \in_R X$	x is a random element of the set X (uniformly distributed)
$X \subset_R Y$	X is a random subset of the set Y (uniformly distributed)
$x \stackrel{?}{=} y$	check whether $x = y$

4.2 Secret Sharing Scheme

Purpose of secret sharing scheme is to share a secret among N authorities in such a way that only some predefined coalitions of authorities can later reconstruct the secret. Other coalitions of authorities should get no knowledge about the secret. We introduce Shamir's $(t+1, N)$ secret sharing scheme from [Sha79] that allows any coalition of $t+1$ from N authorities to get the secret. Any set of at most t authorities knows nothing about the secret.

Let the set of possible secrets forms a field F (for instance, F could be set of real numbers, or Z_p). F should have at least $N+1$ distinct elements – we will denote them $0, 1, 2, \ldots N$.

Distribution of the shares. A secret $s \in F$ is distributed among the N authorities; each authority gets its share $s_j \in F$. The idea behind is simple: Choose a random polynomial f of degree t over the field F satisfying $f(0) = s$. Give the authority A_j its share $s_j = f(j)$.

Reconstruction of the secret. Set of $t+1$ authorities A gains the secret s by reconstructing the polynomial f (using Lagrange interpolation) and computing $s = f(0)$:

$$s = f(0) = \sum_{j \in A} f(j)\lambda_{j,A} = \sum_{j \in A} s_j \lambda_{j,A}$$

$$\lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l-j}$$

Information that t or less authorities have about the polynomial f reveals nothing about the value $f(0) = s$. Whatever value for $f(0) = r$ they choose, using their shares they can compute possible polynomial g satisfying $g(0) = r$.

4.3 Publicly Verifiable Secret Sharing

Publicly verifiable secret sharing scheme is a secret sharing scheme allowing verifying that the dealer has distributed valid shares (any set of $t+1$ authorities will obtain the same secret) and allowing catching the dishonest authority in forging its share. The following publicly verifiable secret sharing comes from [Sch99].

Initialization. The group Z_p and the generators G, g are selected. The authority A_j chooses its secret key z_j and publishes its public key $h_j = g^{z_j}$. The dealer wants to share a secret g^s to the authorities[1].

Distribution of the shares. The dealer picks a random polynomial of degree t over the Z_p:

$$p(x) = \sum_{k=0}^{t} \alpha_k x^k$$

where $\alpha_0 = s$ and $\alpha_1, \ldots, \alpha_t \in Z_p$. The polynomial is kept secret and the commitments $C_k = G^{\alpha_k}$, $0 \leq k \leq t$ as well as the encrypted shares $H_j = h_j^{p(j)}$, $j = 1, 2, \ldots N$ are published. Moreover, the dealer shows that the encrypted shares are consistent: Let $X_j = \prod_{k=0}^{t} C_k^{j^k} = G^{\sum_{k=0}^{t} \alpha_k j^k} = G^{p(j)}$, the dealer proves that

$$\log_G X_j = \log_{h_j} H_j$$

using the non-interactive proof from the section 4.12.2.

Reconstruction of the secret. The authority A_j decrypts its share $S_j = g^{p(j)}$ by computing $S_j = H_j^{1/z_j}$. A_j also proves that $\log_G h_j = -\log_{H_j} S_j$ (again the proof from the section 4.12.2). Further, suppose that $t+1$ authorities A_j, $j \in A$ produce the correct values for S_j, $j \in A$. The secret g^s is reconstructed by Lagrange interpolation

$$\prod_{j \in A} S_j^{\lambda_{j,A}} = \prod_{j \in A} g^{p(j)\lambda_{j,A}} = g^{\sum_{j \in A} p(j)\lambda_{j,A}} = g^{p(0)} = g^s$$

where $\lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l-j}$ is a Lagrange coefficient.

4.4 RSA Cryptosystem

Key Generation. Alice creates her public key and a corresponding private key. Alice should do the following:

1. Generate two large random distinct primes p, q, each roughly the same size.

[1] If the discrete logarithm of the secret $\sigma \in Z_p$ to the base g is not known (i.e. the value $d \in Z_p$ such that $g^d = \sigma$), the dealer chooses s randomly and publishes $U = \sigma g^s$. After the g^s is reconstructed, the secret σ is obtained as $\sigma = U g^{-s}$.

2. Compute $n = pq$, $\phi = (p-1)(q-1)$

3. Select an integer e, $1 < e < \phi$ such that $gcd(e, \phi) = 1$.

4. Compute the unique integer d, $1 < d < \phi$ such that $ed \equiv 1 \bmod \phi$

5. The public key is (n, e), the private key is d.

Encryption. To encrypt an integer m, $0 \leq m < n$, Bob should compute $c = m^e \bmod n$.

Decryption. Alice recovers the plaintext m from the ciphertext c as $m = c^d \bmod n$.

4.5 Quadratic Residuosity Problem

Let n be an integer. Any $y \in Z_n^*$ which can be written as $y = x^2 \bmod n$ for some $x \in Z_n$ is called a *quadratic residue modulo n*. The set of quadratic residues modulo n will be denoted as Q_n.

If $n = p$ is a prime number, the Legendre symbol is defined as

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p|a; \\ 1, & \text{if } a \in Q_p; \\ -1, & \text{if } a \notin Q_p. \end{cases}$$

If n is composite and $n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ is its factorization, the Jacobi symbol is defined as

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \cdots \left(\frac{a}{p_k}\right)^{e_k}$$

There exists efficient algorithm for computing $\left(\frac{a}{n}\right)$ for any a, n (the factorization of the n does not need to be known) – see for instance [MvOV96].

Obviously, if a is a quadratic residue, then $\left(\frac{a}{n}\right) = 1$. However, from $\left(\frac{a}{n}\right) = 1$ does not follow that a is a quadratic residue. Such a which is not a quadratic residue but satisfies $\left(\frac{a}{n}\right) = 1$ is called a *pseudo-square*. The set of pseudo-squares is denoted by $\widetilde{Q}_n$.

If $n = pq$ is a product of two distinct primes, then $|Q_n| = |\widetilde{Q}_n| = \frac{(p-1)(q-1)}{4}$.

Quadratic residuosity problem is the following: given an odd composite integer n and $a \in Z_n^*$ such that $\left(\frac{a}{n}\right) = 1$, decide whether a is a quadratic residue modulo n.

If $n = p$ is a prime number, it is easy to decide whether $a \in Z_n$ is a quadratic residue modulo p or not, since it follows directly from the definition of Legendre symbol and the $\left(\frac{a}{p}\right)$ can be efficiently computed.

If $n = p_1^{e_1} \cdots p_k^{e_k}$ is composite, then a is a quadratic residue modulo n if and only if it is a quadratic residue modulo p_i for all $i = 1, \ldots k$. Thus, if the factorization of n is known, quadratic residuosity problem can be solved by checking whether $\left(\frac{a}{p_i}\right) = 1$ for all $i = 1, \ldots k$. On the other hand, if the factorization of n is not known, then there is no efficient way known for solving

quadratic residuosity problem. If $n = pq$ the correct answer can be guessed with the probability $\frac{1}{2}$. It is believed that quadratic residuosity problem is as difficult as factoring, but no such proof has been given.

More details about quadratic residuosity problem can be found in [MvOV96].

4.6 ElGamal Cryptosystem

ElGamal public-key cryptosystem can be based on any family of groups for which the discrete logarithm is considered intractable. Usually a subgroup G_q of order q of Z_p is used, where p, q are large primes satisfying $q|p-1$. Other practical groups can be obtained for elliptic curves over finite fields. The discrete logarithm problem for elliptic curves is considered to be harder. We present construction in the group Z_p, where p is a large prime.

Key generation. Alice creates a public key and a corresponding private key. Alice should do the following:

1. Generate large prime p and a generator g of the multiplicative group Z_p of the integers modulo p.
2. Select random integer α, where $1 \leq \alpha \leq p-2$, and compute $h = g^\alpha$.
3. The public key is (p, g, h), and the private key is α.

Encryption. Bob obtains Alice's public key (p, g, h). Bob wants to encrypt a message $0 \leq m < p$ for Alice. He should do the following:

1. Select a random integer k, $0 \leq k \leq p-2$.
2. Compute $x = g^k$, $y = mh^k$.
3. Send the ciphertext $c = (x, y)$ to Alice.

Decryption.To recover plaintext m from $c = (x, y)$, Alice should do the following:

1. Using the private key α, compute $r = x^{p-1-\alpha}$. (Note that $r = x^{p-1-\alpha} = x^{-\alpha} = (g^k)^{-\alpha} = g^{-k\alpha}$).
2. Recover m by computing $m = yr \bmod p$.

4.7 Blind Signatures

We require the signatures to be genuine (only the signer can sign messages) and publicly verifiable (anyone can verify whether the given signature of the message is correct).

If the signer has RSA public key (n, e) and the corresponding private key d, he can sign a message m, $m \in Z_n$ as $s = m^d \bmod n$. Given the signature s of the message m, anyone can verify its validity by checking whether $m \stackrel{?}{=} s^e \bmod n$.

Notice that the decryption and encryption methods of the RSA cryptosystem are used in signing the message and verifying its signature.

Suppose that a requester wants to obtain the signer's signature of the message m. The requester does not wish to reveal the message m to anyone, including the signer. The signer is requested to sign a message blindly, not knowing what he signs. [2]

If the signer has RSA public (n, e) and the corresponding private key d, the requester obtains blind signature of the message m as follows:

1. The requester blinds his message m to $m' = mr^e \bmod n$, where $r \in_R Z_n$ is random, and sends m' to the signer.

2. The signer signs the blinded message m' and sends its signature $s' = m'^d \bmod n$ to the requester.

3. The requester retrieves the desired signature s of the message m by computing
$$s = \frac{s'}{r} = \frac{m'^d}{r} = \frac{m^d r^{ed}}{r} = \frac{m^d r}{r} = m^d \pmod{n}$$

Formally, the blind signature scheme with message space M is a 5-tuple $(\eta, \chi, \sigma, \delta, \Gamma)$, where

- η is a poly-time probabilistic algorithm, that constructs the signer's public key (pk) and its corresponding secret key (sk);
- χ is a poly-time blinding algorithm, that on input a message $m \in M$, a public key pk and a random string r, constructs a blind message m';
- σ is a poly-time signing algorithm, that on input a blind message m' and the secret key sk constructs a blind signature s' on m';
- δ is a poly-time retrieving algorithm, that on input a blind signature s' and the random string r extracts a signature s on m;
- Γ is a poly-time signature-verifying algorithm that on input a message-signature pair (m, s) and the public key pk outputs either yes or no.

For threshold blind signatures (where the secret key of the blind signature scheme is shared among the N authorities), see [DK00], or [JL99].

[2] Imagine, for instance, that the voter creates his ballot m and he needs authority's approval and authorization of the ballot, but revealing the ballot m to the authority would compromise his privacy.

4.8 Bit Commitment

Suppose that Alice wants to send a bit b to Bob. She does not wish to reveal b to Bob immediately. Bob requires that Alice cannot change her mind afterwards and that the bit she later reveals will be the same as she thinks of now.

Alice encrypts the bit b in some way and sends the encryption to Bob. Bob is not able to recover b until Alice sends him the key. Encryption of b is called a *blob*. In general, the bit commitment scheme is a function $\xi : \{0,1\} \times X \to Y$, where X, Y are finite sets. An encryption of b is any value $\xi(b,k)$, $k \in X$. Bit commitment scheme should satisfy the following properties:

- *concealing* – Bob cannot determine the value b from $\xi(b,k)$
- *binding* – Alice can later open the $\xi(b,k)$ by revealing b, k used in its construction. She should not be able to open the blob as both a 0 and a 1.

If Alice wants to commit a string of bits, she commits every bit independently.

Bit commitment scheme in which Alice is able to open the blob as both a 0 and a 1 is called a *trapdoor bit commitment* .

Bit commitment can be performed as follows:

Suppose that a large prime p, a generator g of Z_p and a $G \in Z_p$ are known. Discrete logarithm of G to the base g should not be known both to Alice and to Bob (G can be chosen randomly). Bit commitment $\xi : \{0,1\} \times Z_p \to Z_p$ is

$$\xi(b,k) = g^k G^b$$

Let $\log_g G = a$. The blob can be opened as b by revealing k, and can be opened as $\neg b$ by revealing $k - a$ if $b = 0$ or $k + a$ if $b = 1$. If Alice does not know a, she is not able to open the blob as $\neg b$.

Similarly, if Bob does not know k, he cannot determine b seeing just $\xi(b,k) = g^k G^b$.

Trapdoor bit commitment scheme is obtained in the case that a is known to Alice.

If the a is known to Bob and Alice opens the blob to Bob through untappable channel, Bob is able to lie to the third party about the committed bit b. Simply, he claims he has received $k - a$ or $k + a$ instead of k. Bit commitment scheme allowing the verifier (Bob) to lie about the opening of the blob is called *chameleon bit commitment*.

Instead of committing every bit of the string s independently, Alice can simply commit to $0 \leq s < p$ by $\xi(s,k) = G^s g^k$. Again, the knowledge of a gives Alice the ability to open the $\xi(s,k)$ as any s', k' satisfying $as + k = as' + k'$.

4.9 Homomorphic Encryption

Consider a probabilistic encryption scheme. Let P be the plaintext space and C the ciphertext space such that P is a group under the binary operation

$\oplus$ and C is a group under the operation $\otimes$. Instance E of the probabilistic encryption scheme is created by generating its public and private keys. Let $E_r(m)$ denotes encryption of the message m using the random parameter(s) r in the instance E; r is a randomness used in the process of encryption.

We say that the probabilistic encryption scheme is $(\otimes, \oplus)$-*homomorphic*, if for any instance E of the encryption scheme, given $c_1 = E_{r_1}(m_1)$ and $c_2 = E_{r_2}(m_2)$, there exists an r such that

$$c_1 \otimes c_2 = E_r(m_1 \oplus m_2)$$

For example, ElGamal encryption scheme is homomorphic. Here, P is a set of integers modulo p ($P = Z_p$), and C is a set of couples $C = \{(a, b)|a, b \in Z_p\}$. The operation $\oplus$ is multiplication modulo p. For binary operation $\otimes$ defined on cipher texts lets take multiplication modulo p per components. Two plaintexts m_0, m_1 are encrypted to

$$E_{k_0}(m_0) = (g^{k_0}, h^{k_0} m_0)$$

$$E_{k_1}(m_1) = (g^{k_1}, h^{k_1} m_1)$$

where k_0, k_1 are random.

It holds

$$E_{k_0}(m_0) E_{k_1}(m_1) = (g^{k_0} g^{k_1}, h^{k_0} h^{k_1} m_0 m_1) = (g^k, h^k m_0 m_1) = E_k(m_0 m_1)$$

for $k = k_0 + k_1$.

Therefore in ElGamal cryptosystem, by multiplication of ciphertexts we gain encrypted multiplication of corresponding plaintexts.

4.10 Robust Threshold ElGamal Cryptosystem

The purpose of threshold public-key cryptosystem is to share a private key among the authorities such that messages can be decrypted only when a substantial set of authorities cooperate. We need to change the key generation and the decryption protocol in the ElGamal cryptosystem. Messages will be encrypted as usual.

Key generation. The result of the key generation protocol is that each authority A_j will possess a share s_j of a secret s (a private key in the ElGamal cryptosystem) and the public key will be made public. The authorities are committed to their shares as the values $h_j = g^{s_j}$ are published. Furthermore, the shares s_j are such that the secret s can be reconstructed from any set of $t+1$ shares. Any set of at most t shares can tell nothing about the secret s. To achieve this, Shamir's $(t + 1, N)$ secret sharing scheme is used. Trusted third party is needed to compute and distribute these shares to authorities using untappable channel (key generation protocol without the trusted third party is presented in [GJKR99]).

Thus, it holds

$$s = \textstyle\sum_{j \in A} s_j \lambda_{j,A} \quad \lambda_{j,A} = \prod_{l \in A - \{j\}} \frac{l}{l - j}$$

The public key is (p, g, h), where $h = g^s$.

Decryption. To decrypt a cipher text $(x, y) = (g^k, h^k m)$ without reconstructing the secret s, the authorities execute the following protocol:

1. Each authority A_j broadcasts $w_j = x^{s_j}$ and proves in zero-knowledge that

$$\log_g h_j = \log_x w_j$$

   A proof from the section 4.12.2 is sufficient for this purpose; it is made non-interactive using the technique from the section 4.12.1.

2. Let A is any set of $t+1$ authorities who passed the zero-knowledge proof. The plaintext can be recovered as

$$m = \frac{y}{x^s}$$

$$x^s = x^{\sum_{j \in A} s_j \lambda_{j,A}} = \prod_{j \in A} w_j^{\lambda_{j,A}}$$

At most t authorities' secrets s_j can be disclosed, as from the $t+1$ known values s_j a secret key s can be computed (using Lagrange interpolation), and the message can be directly recovered as in ElGamal decryption.

4.11 Mix Nets

Main idea of Mix Nets is to permute and modify (e.g. decrypt or re-encrypt) some sequence of objects in order to hide the correspondence between elements of original and final sequence. David Chaum proposed this idea in 1981 as a realization of anonymous channel (see [Cha81]).

There are n mix-servers $M_1, \ldots M_n$; each with his own public key E_j and private key D_j. When someone wants to send a message m through anonymous channel, he encrypts it

$$E_1(E_2(\ldots E_n(m))\ldots)$$

and sends to M_1.

M_1 waits until more encrypted messages arrive. Then it takes the received messages, it removes one level of encryption, permute them in random order, and sends them to M_2.

Mix-server M_j receives the encrypted messages. It removes one layer of encryption, shuffles them and sends $E_{j+1}(E_{j+2}(\ldots E_n(m))\ldots)$ to M_{j+1}. The last mix-server M_n decrypts the messages and sends them to their recipients.

We can require that mix-server should give a proof of correct decryption and permutation of the messages (see [PIK93]).

4.12 Interactive Proofs

In this section, the interactive proofs used in the voting schemes are presented. These proofs are based on the discrete log assumption and on the

ElGamal cryptosystem. Similar proofs can be designed for other cryptosystems (e.g. Paillier's cryptosystem [DJ01]). All of these interactive proofs can be made non-interactive using Fiat-Shamir technique described in the section 4.12.1. The presented protocols are not known to be zero-knowledge, but they suffice our application.

4.12.1 Making interactive proof non-interactive

All mentioned protocols have similar structure: the prover wants to prove P. He sends some A to the verifier, who gives the prover a challenge C (sequence of random bits) and finally the prover computes a respond $R = respond(P, A, C)$. The communication $(P; A, C, R)$ and the fact that the prover knew nothing about the C at the time he has computed A persuades the verifier that P holds.

The whole protocol will become non-interactive if we order the prover himself to generate unexpected random bits C. The prover should not be able to generate C before he creates P and A. Fiat and Shamir proposed a technique where C is an output of the hash function: $C = H(P, A)$.

The non-interactive proof is constructed as

$$(P; A, H(P, A), R)$$

where $R = respond(P, A, H(P, A))$.

4.12.2 Equality of Discrete Logarithms

In this section, we present protocol that shows equality of discrete logarithms. The prover has an 4-tuple (g, x, h, y), $g, x, h, y \in Z_p$, and he shows possession of an $\alpha \in Z_p$ satisfying $x = g^\alpha$ and $y = h^\alpha$. The protocol is depicted in the figure 2. Security properties of this protocol can be found for instance in [CGS97].

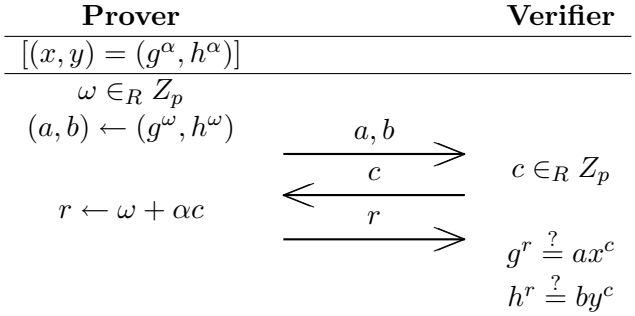

Figure 2: Proof of knowledge for $\log_g x = \log_h y$

For random c, r anyone can construct $(g^r x^{-c}, h^r y^{-c}, c, r)$, which is the accepting conversation with the right distribution. However, the prover sends a, b before he receives the challenge c. Hence, without the knowledge of α he cannot compute the respond r that meets verifier's requirements.

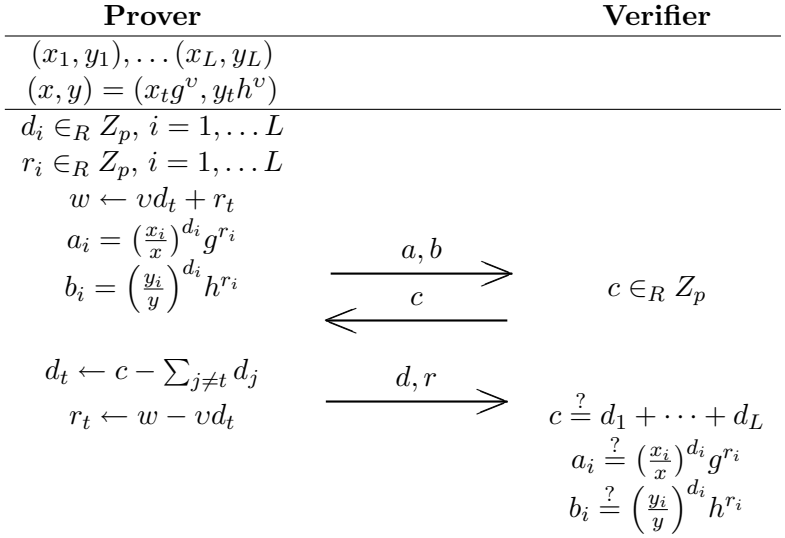

Figure 3: 1-out-of-L re-encryption proof

Non-interactive version

- The prover's computations are the same as in the interactive proof, but he generates the challenge c for himself as $c = H(a\|b\|x\|y)$, where H is a secure hash function. The prover stores c, r as a proof.
- The verification can be performed by checking whether

$$c \stackrel{?}{=} H(g^r x^{-c} \| h^r y^{-c} \| x \| y)$$

Notice that instead of four group elements that are communicated in the interactive protocol, the non-interactive version needs to store only two group elements.

4.12.3 1-out-of-L Re-Encryption Proof

A prover wants to prove that for the encrypted message (x, y) there exists a re-encryption in the L encrypted messages $(x_1, y_1), (x_2, y_2), \ldots, (x_L, y_L)$. The messages are encrypted using ElGamal cryptosystem.

Assume that the re-encryption of (x, y) is (x_t, y_t) and that the re-encryption randomness (the witness) is υ, i.e. $(x_t, y_t) = (xg^{\upsilon}, yh^{\upsilon})$. For the ease of understanding, the protocol is depicted in the figure 3. Note that the a, b, d, r from the protocol are vectors: $a = (a_1, \ldots, a_L)$, $b = (b_1, \ldots, b_L)$, $d = (d_1, \ldots, d_L)$ and $r = (r_1, \ldots, r_L)$.

The sent values a_i, b_i commit the prover to d_i and r_i for all $i = 1, 2, \ldots, L$, except for $i = t$. Values a_t and b_t only commit the prover to a value $w = \upsilon d_t + r_t$, since $a_t = g^{\upsilon d_t + r_t}$ and $b_t = h^{\upsilon d_t + r_t}$. As the prover knows υ, he can still change d_t and r_t after this round.

The verifier challenges the prover to modify his d and r such that d sum to the random number c. Actually, the prover modifies the values d_t, r_t to satisfy those requirements ($c = d_1 + d_2 + \cdots + d_L$ and $w = vd_t + r_t$), and sends (modified) $d_1, d_2, \ldots d_L$ and $r_1, r_2, \ldots d_L$ to the verifier. This ability persuades the verifier that among L encrypted pairs really is one re-encryption of (x, y) and that the prover knows the re-encryption randomness; otherwise he could not adapt his values to the given sum.

For security properties of this protocol, see [CGS97].

Non-interactive version

- The prover's computations are the same as in the interactive proof, but he generates the challenge c for himself as

  $$c = H(a_1 \| \ldots \| a_L \| b_1 \| \ldots \| b_L \| x \| y \| x_1 \| \ldots \| x_L \| y_1 \| \ldots \| y_L)$$

  where H is a secure hash function. The prover stores $c, d_1, \ldots, d_L, r_1, \ldots, r_L$ as a proof.

- The verification can be performed by checking whether

  $$c \stackrel{?}{=} H(a_1 \| \ldots \| a_L \| b_1 \| \ldots \| b_L \| x \| y \| x_1 \| \ldots \| x_L \| y_1 \| \ldots \| y_L)$$

  where

  $$a_i = \left(\frac{x_i}{x}\right)^{d_i} g^{r_i}$$

  $$b_i = \left(\frac{y_i}{y}\right)^{d_i} h^{r_i}$$

Notice that instead of $4L + 1$ group elements that are communicated in the interactive protocol, the non-interactive version needs to store only $2L+1$ group elements.

4.12.4 L Possibilities for Discrete Logarithm

For the encrypted message $(x, y) = E(m)$ (in ElGamal cryptosystem) we want to give a proof that m is one from L possible messages $G_1, \ldots G_L$ and nothing else. Revealing any information about the m except that it really belongs to this set is not desired. We suppose that the discrete logarithms (to the bases g, h) of the elements $G_1, \ldots G_L$ are not known.

The aim is to prove that

$$\log_g x = log_h(y/G_1) \vee log_g x = \log_h(y/G_2) \vee \cdots \vee log_g x = log_h(y/G_L)$$

It is enough to show that among the elements

$$\begin{aligned}(x_1, y_1) &= (x, y/G_1)\\ (x_2, y_2) &= (x, y/G_2)\\ &\vdots\\ (x_L, y_L) &= (x, y/G_L)\end{aligned}$$

is the re-encryption of $(1, 1)$. For this we can use the protocol from the section 4.12.3.

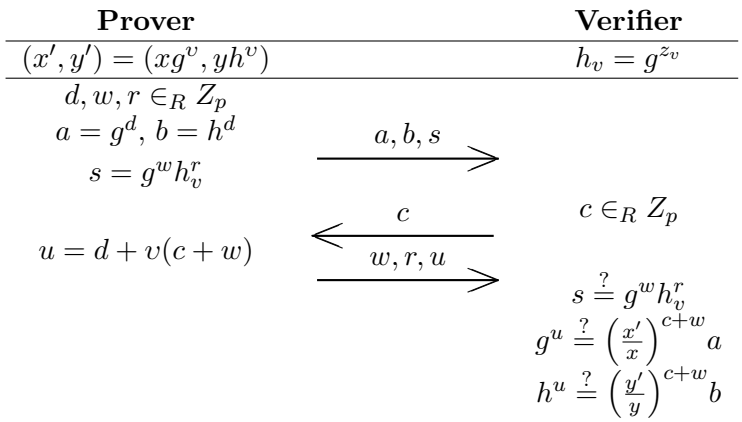

Figure 4: Designated-verifier re-encryption proof

4.12.5 Designated-Verifier Re-Encryption Proof

The prover wants to privately prove that (x', y') is the re-encryption of $(x, y) = (g^k, h^k m)$, i.e. that $(x', y') = (xg^{\upsilon}, yh^{\upsilon})$, where υ is the re-encryption randomness. The proof is constructed for the particular verifier who possess a secret z_{υ} (discrete logarithm of h_{υ} to the base g_{υ}; $h_{\upsilon} = g^{z_{\upsilon}}$). Knowledge of z_{υ} enables him to construct this kind of proof for any couples (x', y'), (x, y). The protocol relies on the verifier's knowledge of z_{υ}. If this property is not ensured by the underlying public-key infrastructure, then the protocol, which ensures the knowledge of the secret key (section 4.12.6), is performed.

The interactive proof is depicted in the figure 4. Values a, b, s sent to the verifier commits the prover to d, w, r. The prover is not able to change the values w, r. However, the verifier can use his knowledge of z_{υ} to open s to arbitrary values w', r' satisfying $w + rz_{\upsilon} = w' + r'z_{\upsilon}$.

Non-interactive version

1. The prover's computations are the same as in the interactive proof, but he generates the challenge c for himself as $c = H(x\|y\|x'\|y'\|a\|b\|s)$, where H is a hash function, and stores c, w, r, u as a proof.

2. The verification can be performed by checking whether

$$c \stackrel{?}{=} H\left(x\|y\|x'\|y' \parallel \frac{g^u}{\left(\frac{x'}{x}\right)^{c+w}} \parallel \frac{h^u}{\left(\frac{y'}{y}\right)^{c+w}} \parallel g^w h_{\upsilon}^r\right)$$

Notice that instead of 7 group elements that are transferred in the interactive protocol, the non-interactive version stores only 4 group elements.

How can the verifier forge the proof. The verifier who knows the secret z_{υ} such that $h_{\upsilon} = g^{z_{\upsilon}}$ can generate the non-interactive proof for any (x, y) and

(x^*, y^*). The key point is that the value s does not commit the verifier to w and r. The verifier selects α, β, u^* at random, sets $E = x\|y\|x^*\|y^*$ and computes

$$c^* = H\left(E\|\frac{g^{u^*}}{\left(\frac{x^*}{x}\right)^{\alpha}}\|\frac{h^{u^*}}{\left(\frac{y^*}{y}\right)^{\alpha}}\|g^{\beta}\right)$$

$$w^* = \alpha - c^*$$

$$r^* = \frac{\beta - w^*}{z_v}$$

and sets the (c^*, w^*, r^*, u^*) as the proof.

4.12.6 Ensuring the Knowledge of the Secret-Key

The following protocol is used to verify that the voter really knows his secret key z_v corresponding to the public key $h_v = g^{z_v}$. Even if the voter does not know his secret key and he acts according to the coercer's orders (the coercer knows the secret-key), he finally gets to know his secret key.

The voter's knowledge of the z_v is verified by the N authorities. It is assumed that at least t of them are honest. The untappable channel between the voter and the authorities is needed.

1. The voter shares his secret key z_v among the authorities using $(t+1, N)$ secret sharing scheme:
   - He chooses a random polynomial of degree t: $f_v(x) = z_v + \alpha_1 x + \cdots + \alpha_t x^t$
   - He sends $s_j = f_v(j)$ through the untappable channel to the authority A_j, $j = 1 \ldots N$
   - He commits to the coefficients of the polynomial by sending $C_j = g^{\alpha_j}$ to the bulletin board
2. Each authority A_j verifies whether the received share s_j corresponds to the committed polynomial

$$g^{s_j} \stackrel{?}{=} h_v C_1^j C_2^{j^2} \cdots C_t^{j^t} (= g^{z_v} g^{\alpha_1 j} g^{\alpha_2 j^2} \cdots g^{\alpha_t j^t} = g^{f_v(j)})$$

3. If the authority A_j detects an error, it complains and the voter is asked to publish its share to the bulletin board. If the posted share does not correspond to the commitments, the voter is discarded.
4. Finally, every authority not complaining in the previous stage sends her share through the untappable channel to the voter.

At least t honest authorities either complain (and their shares are published in the bulletin board), or send their shares secretly to the voter. The voter can interpolate the received shares to obtain the secret key z_v.

5 Existing Voting Schemes

Despite extensive work on the voting schemes, no complete solution has been found in either theoretical or practical domains. A number of practical voting schemes have been proposed, with widely differing security properties.

We do not present all proposed voting schemes due to space limitation. We focus only on the most important ones of them – the milestones in the electronic voting, the schemes introducing new ideas and the schemes efficient in practice. We try to illustrate the progress done both in the security properties as well as in the effectiveness.

The first schemes used anonymous channel for casting the ballots. Later, the schemes exploiting homomorphic encryption were introduced. Next, the possibility for coercing and vote buying in all of these schemes was highlighted, and the ways to achieve receipt-freeness were developed.

Schemes using anonymous channel and blind signatures are very popular in practice due to their efficiency and their support for any type of the voting. A price is paid for this efficiency: the voter has to act in more rounds (registration, voting, counting, verifying whether his vote has been counted, complaining...) and they provide no universal verifiability (usually).

Schemes using homomorphic encryption have more security properties, but their communication complexity is quite high. Privacy is protected by the encryption method. Coalition of all authorities usually can decrypt the voter's vote and violate the privacy. In addition, these schemes do not support any type of the voting. They were designed to yes-no voting or 1-out-of-L voting, and can be extended to K-out-of-L voting or 1-L-K voting.

5.1 The First Voting Scheme

The anonymous channel and the first voting scheme was proposed by Chaum [Cha81].

The authorities are N mix-servers with their public keys $E_1, \dots E_N$. The voter V_i generates his public key K_i and writes $E_1(E_2(\cdots E_N(K_i))\cdots)$ to his section in the bulletin board. Mix-servers shuffles these messages (sequentially permute and decrypt, see section 4.11), and produce a list of keys K_i. Here, the voter may claim when his K_i is not on the list. In that case, the elections are restarted. If no complain is raised, the voter writes $E_1(\cdots E_N(K_i \| K_i^{-1}(v_i))\cdots)$ in the bulletin board. Again, the mix-servers shuffles these messages, and the list of $K_i \| K_i^{-1}(v_i)$ is combined with the previous list to obtain the votes v_i.

This scheme has many drawbacks. For example, failure of the single voter will disrupt the election and the election has to be restarted. Moreover, if the election has to be restarted after the second phase, when some votes have already been published, it can affect the re-election. This scheme has been enhanced in fairness and efficiency of the anonymous channel in [PIK93].

5.2 Schemes Based on Blind Signatures and Anonymous Channel

The schemes from [Cha88, Boy, FOO92, Rad95, JL97, JLY98] are of this kind. Roughly said, these schemes works in the following way: The voter firstly

obtains a token – in fact, the token is a message blindly signed by the authority. The voter is able to obtain only one token, since the authority blindly signs only one message for the voter. Next, the voter sends the token with his vote through the anonymous channel back to the authority. The authority collects the votes and publishes them together with the tokens.

The authority issuing the tokens will be called an administrator, and the authority collecting the votes a collector. The terms administrator and collector can refer to the two different authorities, or to the same authority acting both as the administrator and as a collector, depending on the scheme.

This approach brings about some problems and security drawbacks that should be solved somehow in the voting schemes. The most important ones are as follows:

- no fairness – some participant (the collector) knows the intermediate result (partial sum) before the counting stage
- not collision-free – there is a chance that two voters will gain the same token at the registration, hence the vote of one voter will be excluded as double-vote
- a dishonest authority (administrator) may impersonate the voters abstaining from the voting and add its own votes, or secretly provide some voters with more than one token
- in the case that the voter's vote has not been counted, the voter cannot complain without revealing his vote

To enhance the fairness, we can prevent the collector to see the actual votes with a simple trick. Just let it collect the encryptions of the votes. The actual votes will be decrypted later at the counting stage. The decryption key can be sent anonymously by the voter (who has encrypted the vote), as it is in the FOO-scheme (section 5.2.1), or the decryption key can be reconstructed by some set of authorities (for instance, the JL-scheme from the section 5.2.3).

Collision-freeness can be achieved by inserting the voter's identification into the token in such a way that it is infeasible to extract it.

A dishonest behavior of the authority may be avoided by distributing the power of the single authority to several authorities.

First scheme based on blind signatures was [Cha88]. It supports only 1-out-of-L voting; it has large communication complexity at the registration phase (proportional to ML); it is not fair and not collision-free.

Boyd's scheme from [Boy]) is more efficient, but it is also not fair and not collision-free.

The first practicable scheme ensuring both the privacy and the fairness is of Fujioka, Okamoto and Ohta (section 5.2.1). It also prohibits the fraud by either the voter or the authority. The voter has to participate in three rounds and he has to send two messages through anonymous channel. Reduced number of rounds appear in the scheme of Juang and Lei (section 5.2.3).

The scheme of Radwin (section 5.2.2) presents an idea how to trace double-voters (this idea is also used in electronic cash).

5.2.1 FOO-Scheme

This scheme comes from [FOO92].

Two authorities, an administrator and a collector, manage the elections. The administrator is responsible for token issuing; the collector collects the votes and publishes the result of the election. Token is an encrypted vote, (blindly) signed by the administrator. The voter sends his token anonymously to the collector. The collector collects the tokens, numbers them, and publishes the list of the tokens at the end of the election. The voter looks up his token in the list, and sends his token number together with the key anonymously to the collector. The collector publishes the keys, decrypts the votes and announces the result of the election.

Let ID_i be the identification of the voter V_i, σ_i be the V_i's signature scheme, and σ_A the signature scheme of the administrator. Further, χ is a blinding and δ a retrieving technique used in blind signatures (section 4.7).

Initialization stage. Administrator generates its signature scheme and publishes the public key.

Registration phase. Voter prepares his ballot as follows:

- V_i selects his vote v_i and creates the ballot $x_i = \xi(v_i, k_i)$, where ξ is a secure bit-commitment using the random key k_i (for bit-commitment, see section 4.8).
- V_i computes the message e_i using the blinding technique $e_i = \chi(x_i, r_i)$
- V_i signs $s_i = \sigma_i(e_i)$ and sends (ID_i, e_i, s_i) to the administrator.

Administrator A receives (ID_i, e_i, s_i) and checks whether:

- Voter V_i has the right to vote.
- V_i has not yet applied for the signature.
- The signature s_i of the message e_i is valid.

If all these conditions are satisfied, then the administrator A signs $d_i = \sigma_A(e_i)$ and sends d_i to the voter. If any of these conditions does not hold, the administrator rejects the signature.

At the end of the registration phase, the administrator announces the number of voters who where given the administrator's signature, and publishes the list (ID_i, e_i, s_i).

Voting phase.

- Voter V_i retrieves the desired signature y_i of the ballot x_i by retrieving technique $y_i = \delta(d_i, r_i)$.
- V_i checks that y_i is the administrator's signature of the x_i. If the check fails, V_i claims the disruption by showing that (x_i, y_i) is invalid.
- V_i sends his token (x_i, y_i) anonymously to the collector.

- The collector C checks the administrator's signature y_i of the ballot x_i. If the check succeeds, C enters (l, x_i, y_i) onto a list as an l-th item.

Counting stage. Counting stage consists of two phases: Opening and Counting.

Opening phase. When all of the voters had voted, the collector C publishes the list (l, x_i, y_i). Voter V_i then do the following:

- V_i checks that the number of ballots in the list is equal to the number of voters. If the check fails, voter claims this by revealing the token x_i, y_i and the blinding factor r_i.
- V_i checks that his ballot is listed on the list. If his vote is not listed, then V_i claims this by revealing (x_i, y_i), the valid ballot and its signature.
- V_i sends the key k_i with number l, i.e. (l, k_i) to C through anonymous channel.

Counting phase.

- The collector C opens the commitment of the ballot x_i, retrieves the vote v_i, adds k_i and v_i to the list, and checks that the v_i is a valid vote.
- C counts the votes and publishes the voting result.

Achieved Properties

Eligibility. Only eligible voters are allowed to gain the token. Invalid tokens and invalid votes will be detected. The token cannot be used multiple times, so the voter can vote at most once. Therefore, the eligibility is achieved.

Privacy. The voter's privacy is preserved even if the administrator and the collector conspire: the relation between the voter's ID_i and his ballot x_i is hidden by the blind signature scheme. The voter sends his ballot x_i as well as the key k_i through anonymous channel, so no one can trace it back.

Individual Verifiability. The scheme is individually verifiable: the voter can check whether his ballot (x_i, y_i) is on the list published by the collector, and whether his k_i, v_i has been added to the list.

When the voter claims the disruption, he need not to release his vote v_i, only x_i, y_i. The claiming voters showing valid tokens can be allowed to register for the second time to obtain the new tokens. By any means, at the counting stage when the opening of the tokens are published, everybody gets to know which votes were sent by the claiming voters. Unless the election is restarted when a valid claim occurs, the privacy of the claiming voters is violated.

Universal Verifiability. The scheme is not universally verifiable – if some voters abstain from voting after the registration phase, the administrator can add its own votes instead of theirs. The voter has to participate in three rounds: registration, voting and opening.

Fairness. This election scheme is fair – counting of the ballots does not affect the voting, as the counting stage comes after the voting phase.

Receipt-Freeness. Anyone who gets to know the voter's token can easily find out his vote in the list published by the collector at the end of the election. Therefore, the receipt-freeness is not achieved.

A Modification Achieving Receipt-Freeness

A [Oka97] is in fact a modification of the FOO-scheme guaranteeing incoercibility. The authorities managing the election are: an administrator (registration manager), a collector (collecting the tokens) and a counter (publishing the actual votes and the final tally). The list of changes is as follows:

- ξ is a trapdoor bit commitment
- It is ensured that the voter knows the trapdoor of the bit commitment ξ.
- The voter sends his token (x_i, y_i) anonymously to the collector.
- The voter sends the opening (v_i, k_i, x_i) of the x_i through the untappable anonymous channel to the counter.
- The collector publishes the list of the tokens (x_i, y_i).
- The counter publishes the actual votes v_i in random order and proves that he knows the permutation π and k_i such that $x_{\pi(i)} = \xi(v_i, k_i)$ without revealing π, k_i (π maps the votes to the tokens).

As the trapdoor bit commitment ξ does not bind the voter to the vote v_i, the coercer has no information about his actual vote seeing his token (x_i, y_i). The coercer cannot intercept the opening of the ballot, as it is sent through untappable channel. The counter keeps the opening keys k_i as well as the permutation π mapping the votes to the tokens secret. The counter cannot publish any votes it wants, as the proof of knowledge is required.

The coercer may order the voter to use a bit commitment scheme without trapdoor (e.g. the coercer generates $G = g^a$ for the voter who now do not know a). Therefore, it should be ensured that the voter knows the trapdoor and that he is really able to open the token in various ways. For more details see [Oka97].

The voter may object if his token (x_i, y_i) is not on the list. Privacy of the claiming voters is protected against the public, but not against the counter.

After the registration, the voter cannot abstain from the voting. The administrator can add extra tokens, or the counter may claim it has not received the

opening of the ballot (x_i, y_i). We can avoid this foul behavior by distributing the power of a single administrator and a single counter to the more administrators and the more counters.

5.2.2 Radwin-Scheme

This scheme comes from [Rad95].

We present this scheme with the one reliable authority acting both as the administrator and as the collector. Of course, the scheme can be extended to incorporate more authorities in the election management for the higher security.

Unlike the FOO-scheme, the token does not contain the actual vote. The voter trying to vote twice will be traced. The scheme requires existence of the anonymous channel supporting replays (recipient of the anonymous message can send a replay to the anonymous sender).

We assume the existence of two-argument functions f, g that are collision-free (finding two inputs that maps to the same output is infeasible), output of f looks random and g is 1 to 1 (or c to 1) with the first argument fixed.

Initialization stage. The authority creates and publishes its RSA public key (n, e) and the security parameter l.

Registration phase. The voter V with the identification ID constructs his pseudonym (token) P. For the ease of understanding, the protocol is depicted in the figure 5.

The voter selects numbers $a_k, c_k, d_k, r_k, k = 1, 2, \ldots 2l$ randomly from Z_n and computes $B_k = r_k^e f(x_k, y_k)$ where $x_k = g(a_k, c_k)$, $y_k = g(a_k \oplus ID, d_k)$. ($B_k$ is the blinded message $f(x_k, y_k)$). He sends $B_k, k = 1 \ldots 2l$ to the authority.

There is no reason for the authority to believe that the voter has constructed $B_k, k = 1 \ldots 2l$ as described above. Therefore, the voter is asked to open one half of B_k, randomly selected by the authority. (We denote the set of selected B_k as R). The voter opens B_k by revealing the numbers a_k, c_k, d_k, r_k used in the construction. The authority checks whether revealed values really fit B_k: it verifies whether $B_k = r_k^e f(g(a_k, c_k), g(a_k \oplus ID, d_k))$ for all $k \in R$. If the check succeeds, the remaining half of B_k can be considered to be correctly formed as well. Otherwise, the authority rejects the registration.

Further, the authority signs the remaining $B_k, k \notin R$ by computing $S_k = B_k^d, k \notin R$, where d is its private RSA key, and sends to the voter $S = \prod_{k \notin R} S_k$. Notice that S_k is of the form $S_k = B_k^d = (r_k^e f(x_k, y_k))^d = r_k^{ed} f(x_k, y_k)^d = r_k f(x_k, y_k)^d$ and therefore $S = \prod_{i \notin R} r_k f(x_k, y_k)^d$.

Finally, the voter computes his pseudonym as $P = \prod_{k \notin R} f(x_k, , y_k)$ and its signature $SP = \frac{S}{\prod_{k \notin R} r_k} = \prod_{k \notin R} f(x_k, y_k)^d$.

The set of a_k, c_k, d_k, $k \in R$ is no more useful. For simplicity, we will denote the remaining elements a_j, c_j, d_j, $j \notin R$ as $a_1, c_1, d_1, \ldots a_l, c_l, d_l$.

Voting phase. The voter chooses his vote v, creates a ballot (v, P, SP), encrypts it with the authority's public key (e, n) and sends $(v, P, SP)^e \bmod n$ through the anonymous channel.

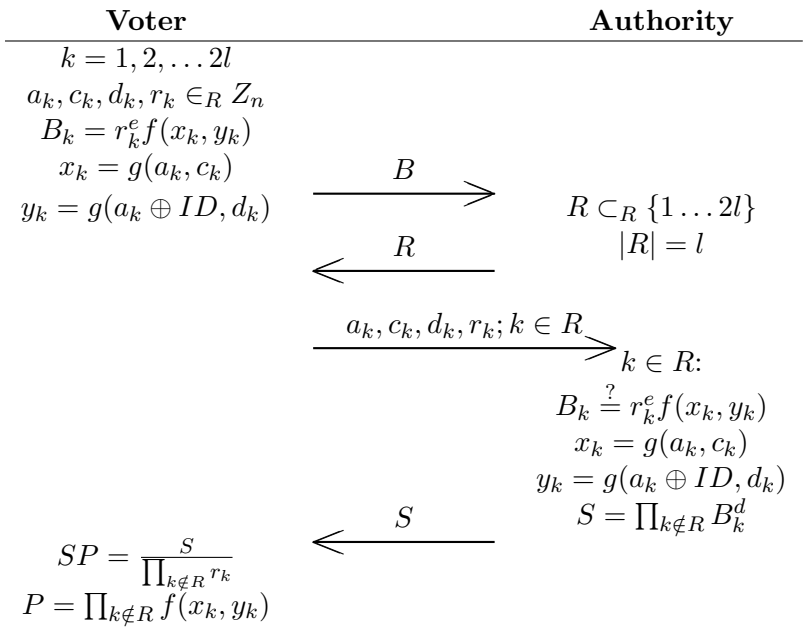

Figure 5: Registration phase – constructing a pseudonym

The authority decrypts this message and verifies the signature SP for the pseudonym P. It sends back to the voter a random binary vector $Z = (z_1, \ldots z_l)$ of the length l.

The voter responds with l triples partially opening the structure of his pseudonym. If $z_k = 0$, the kth triple is a_k, c_k, y_k. If $z_k = 1$, the kth triple is $x_k, a_k \oplus ID, d_k$. That is, for each k one argument of the f is revealed directly, and the second argument can be computed as an output of the g taking the other two revealed numbers as the arguments.

The authority verifies that the voter's responses perfectly fit the P he sent in.

If the test of the pseudonym P succeeds, the authority has not yet received another vote from the voter with pseudonym P and the vote v is valid, the authority counts the vote v.

In the case that the pseudonym P has been used twice, the authority can determine with high probability the identity of this double-voter. When the voter sent a pseudonym P for the first (second) time, he was challenged by a random vector Z_1 (Z_2) to partially reveal the structure of P. With high probability, the binary vectors Z_1 and Z_2 differs at least in one index k. The kth triples of the voter's responses to Z_1, Z_2 now contains a_k and $a_k \oplus ID$. The ID (voter's identification) can be now easily extracted.

Counting stage. The authority counts the received valid votes and publishes the final sum. Two variants of the counting stage are possible:

- No-List Variant: Only the final sum of the votes is made public.

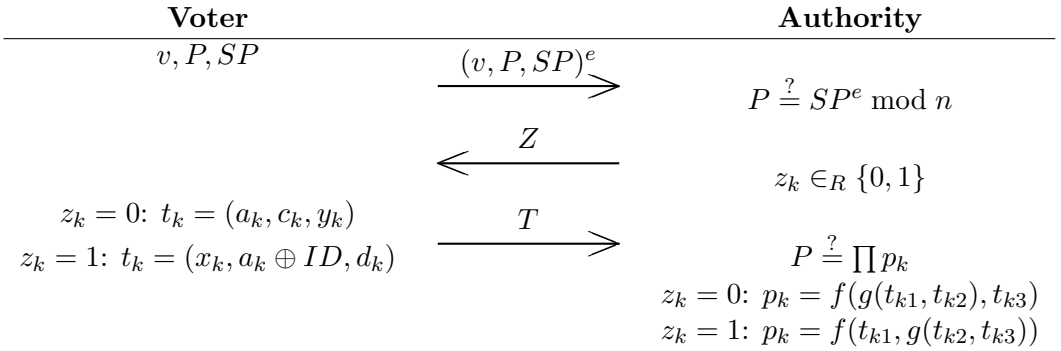

Figure 6: Voting phase – casting a ballot

- List Variant: The authority publishes a list containing received ballots $(v, P, SP)^e$, pseudonyms P, SP, the challenges Z with the answers T and corresponding votes v.

Achieved Properties

Eligibility. The voter cannot construct his pseudonym by himself; he needs the authority's signature. The authority grants the voter only one pseudonym. If the voter tries to use his pseudonym twice, his identity is revealed with high probability. The eligibility is achieved if the authority is honest.

Privacy. At the end of the registration phase, the authority can tell nothing about the voter's pseudonym. The voter discloses only one half of the B_k to the authority, which are not used in the pseudonym. Since the functions f, g are collision-free, the voter cannot generate any other $a'_k, b'_k, c'_k, d'_k, r'_k$ such that $B_k = (r'_k)^e f(g(a'_k, c'_k), g(b'_k, d'_k))$. This way, the authority ensures that the pseudonym is correctly constructed (e.g. the voter has included his ID). As the output of f looks random, the pseudonym (a multiplication of the outputs of f) looks random as well. The relation between the voter and his pseudonym is protected by the blind signature scheme.

The voter's privacy is protected unless he tries to use his pseudonym twice. The ballot cannot be traced back to the voter, since it is sent through anonymous channel.

When the voter uses his pseudonym for the first time, he partially reveals its structure to the authority. The voter can open his pseudonym only in the way it had been constructed, since the functions f, g are collision-free. That is, for the kth output of f he reveals triple t_1, t_2, t_3 where $t_3 = g(t_1 \oplus ID, t_4)$ and the t_4 remains undisclosed. As the g is 1 to 1 (or c to 1) with the first argument fixed, the authority can guess nothing about the $t_1 \oplus ID$ seeing just t_1, t_3. Hence, nothing can be told about the voter's ID.

When the voter attempts to use his pseudonym for the second time, the information he revealed at the first and the second use can be matched to extract his ID.

Individual verifiability. In the No-List Variant only the authority sees the pseudonyms. Hence, nobody can verify which votes the authority has really counted. It can publish any number it wants.

The List Variant is individually verifiable: the voter can verify whether his token and his vote are on the list. If the voter does not find his pseudonym in the list, he can protest by showing a proof that he has sent a valid ballot and that his response to the authority's challenge has been correct. Such a complaint can compromise his privacy (at least the authority will get to know his vote).

Universal verifiability is not achieved, as the authority can impersonate the abstaining voters and add its own votes, or provide some voters with more tokens at the registration phase.

Receipt-Freeness. List Variant is not receipt-free; voter's receipt are the random numbers a_i, c_i, d_i, r_i generated at the registration stage, from which his pseudonym can be constructed and his vote can be retrieved from the published list.

The No-List Variant is receipt-free, if the voter sent his ballot through the anonymous untappable channel. The coercer can obtain the voter's pseudonym, but if he cannot tamper on the anonymous channel, and if the authority will not dispose the list of v, P to anyone, then in the No-List Variant with anonymous untappable channel he has no way to gain the relation between the v and P.

5.2.3 JL-Scheme

This scheme [JL97] involves one manager (acting both as the administrator and as the collector), and N scrutineers. The role of the scrutineers is to share the threshold ElGamal key used in encrypting the votes. Encrypted vote is a part of the token.

This scheme require the valid actual vote to be of the form

$$\bar{v_i} = v_i \| R_i \| g(v_i \| R_i \| RD)$$

where v_i is the intention of the voter V_i, g is a one-way function, R_i are random bits generated by the voter, and RD is a voting tag – redundancy bits that characterize the election. RD is determined and published by the manager at the initialization stage and has to be unique for each election.

RD prevents the voter to re-use the vote from the previous elections. As the vote is a part of the token, the tokens of the previous election are no more useful. The manager does not need to re-generate its signature scheme before each election.

We present the scheme in a simplified form. For more details, see [JL97].

Initialization stage. The manager generates his RSA key (n, e) (for blind signatures), a public one-way permutation f, a public one-way function g and the public redundancy bits RD for verifying the validity of each ballot. The scrutineers set up (t, N) threshold ElGamal cryptosystem with public key (p, g, h).

Registration phase. The voter V_i with identification ID_i chooses random string r_i, s_i and constructs the token

$$x_i = f(ID_i \| s_i) \| RD \| EV_i$$

where $EV_i = (g^{k_i}, h^{k_i}\ \bar{v_i})$, $k_i \in_R Z_p$, is the encrypted actual vote $\bar{v_i}$. The voter blinds his token $e_i = r_i^e x_i$ and sends the blinded message e_i to the administrator.

The manager checks that the voter V_i has not yet registered. If not, the manager sends $d_i = e_i^d$ to the voter.

Voting phase. The voter extracts the signature of his token $y_i = d_i / r_i = x_i^d$ and sends (x_i, y_i) anonymously to the manager. The manager checks whether $y_i^e = x_i$. If yes and the redundancy bits RD from x_i are valid, he records (x_i, y_i) and preserves only one copy of x_i.

Counting stage. The manager publishes all accepted ballots (x_i, y_i). Each voter has to check if his ballot has been published. If his ballot is omitted, he makes an open objection by broadcasting (x_i, y_i).

The manager requests $t + 1$ honest scrutineers to send their secret shares. He computes the secret threshold key and recovers voters' intentions. The administrator publishes all ballots (x_i, y_i, v_i), all registrations e_i and the threshold secret key. Anyone can check if every ballot is valid and the total number of the ballots equals to the total number of the registrations to prevent the administrator from adding extra ballots.

Achieved Properties and Extensions

Eligibility. Each voter can obtain only one token. Invalid tokens or double tokens will be excluded. Invalid votes will not be counted. Therefore, the eligibility is achieved.

Privacy. The link between the ballot (x_i, y_i) and the voter is protected by the security of the RSA signature scheme. Extracting ID_i from the ballot is also computationally infeasible, as f is the one-way function. The sent ballot (x_i, y_i) cannot be traced back to the voter, as it was transmitted through the anonymous channel.

Collision-Freeness. The signed tokens requested by the honest voters are distinct. Therefore, the scheme is collision-free. A dishonest voter V_i may construct a token with $H_i = f(ID_j \| RD)$ $(j \neq i)$, but there is still a small probability that he will obtain the same token as the honest voter V_j.

Individual Verifiability. The scheme is individually verifiable: The voter can check whether his token is on the list.

The voter's complaint, unless sent anonymously, violate his privacy, as it discloses the link between him and his ballot (x_i, y_i).

Abstaining. The voter cannot abstain from the voting after the registration phase. However, the scheme can be revised to distribute the power of a single administrator to several administrators (see [JLY98]). Small coalition of administrators cannot impersonate the abstaining voter, as it is not able to generate valid signature.

Receipt-Freeness. This scheme is not recept-free, since the voter's receipt is his token.

Extension. The power of the manager is distributed among more managers in [JLY98].

5.3 Schemes Using Homomorphic Encryption

The first scheme using homomorphic encryption has been proposed by Benaloh and Yung [Ben87, BY86]. Iversen [Ive91] was inspired by this work. Sako and Kilian [SK94] improved the communication complexity of the Benaloh's scheme.

Big progress was done by Cramer, Gennaro and Schoenmakers in [CGS97]. This efficient and simple scheme (presented in the section 5.3.3) influenced further development in this area.

[Sch99] is less efficient than [CGS97], but it is more suitable for small boardroom election. No interaction between the authorities is needed (except for the setting up the system at the initialization stage)

Receipt-freeness was introduced by Benaloh and Tuinstra [BT94]. They proposed a receipt-free scheme based on the voting-booth. Hirt and Sako in [HS00] point out that their scheme is not receipt-free.

Lee and Kim in [LK00] achieve receipt-freeness by a cooperation of a voter and a honest verifier. The honest verifier helps the voter to encrypt his vote and to prove its validity. However, a coalition of the honest verifier and a voter can cast invalid or double votes.

Hirt and Sako [HS00] (section 5.4) proposed efficient receipt-free scheme based on the work done in [CGS97].

5.3.1 Benaloh's Scheme

The scheme comes from [Ben87]. The scheme introduces many new ideas that were used later in other schemes.

In this scheme, two primitives are used: secret sharing scheme and probabilistic public-key encryption with homomorphic property

$$E(m_1, k_1)E(m_2, k_2) = E(m_1 + m_2, k_1 k_2)$$

(k_1, k_2 are random parameters used in encrypting messages m_1, m_2).

Bulletin board is implicitly used for collecting votes.

Yes-No voting scheme

Initialization stage. Each authority A_j generates its own public key pair. Encryption of a message m with the public key of the authority A_j will be denoted as $E_j(m)$.

Voting stage. Voter cast a vote 0 or 1 as follows:

1. Voter generates a pair (v_1, v_2) as a permutation of $0, 1$.
2. For each $i = 1, 2$ voter creates shares $s_1(v_i), \ldots, s_N(v_i)$ using Shamir $(t+1, N)$ secret sharing scheme with the secret v_i.
3. Voter encrypts jth share with the public key of the authority A_j. He gets

$$h = (h_1, h_2)$$

$$h_i = (h_{i1}, h_{i2}, \ldots, h_{iN}),\ i = 1, 2$$

$$h_{ij} = E_j(s_j(v_i), k_{ij}),\ j = 1, \ldots, N$$

4. Voter publishes h and proves to the authorities (see below) that it is correctly constructed.
5. Voter chooses h_1 or h_2 as his desired vote.

The voter proves the correctness of his vote by the following interactive proof:

1. Voter creates T more pairs of the encrypted votes $h^1, \ldots, h^T$, $h^r_{ij} = E_j(s^r_j(v^r_i), k^r_{ij})$, $1 \le r \le T$, and sends them to the authorities.
2. Authorities generate T random bits $c_1 \ldots, c_T$ and send them to the voter.
3. For each $c_r = 0$ the voter reveals how the pair $h^r = (h^r_1, h^r_2)$ has been constructed by sending $v^r_i, s^r_j(v^r_i), k^r_{ij}$, $i = 1, 2$, $j = 1, \ldots, N$.

   For each $c_r = 1$ he reveals permutation that maps (v_1, v_2) to the (v^r_1, v^r_2). Assume that $v_1 = v^r_1$ and $v_2 = v^r_2$. It holds $h_{ij} = E_j(s_j(v_i), k_{ij})$ $h^r_{ij} = E_j(s^r_j(v_i), k^r_{ij})$. From the homomorphic property it follows that h/h^r is the encryption of the $(0, 0)$-vote: $h_{ij}/h^r_{ij} = E_j(s_j(v_i) - s^r_j(v_i), k_{ij}/k^r ij)$. Voter sends $s_j(v_i) - s^r_j(v_i), k_{ij}/k^r_{ij}$ to the authorities.
4. The authorities check whether the voter's response really fit the h:

   For all $c_r = 0$ they check that h^r is correctly formed.

   For all $c_r = 1$ they check whether h/h^r is the encryption of $(0, 0)$ vote.

Counting stage. Let voter V_i's vote be $h_i = (h_{i1}, \ldots, h_{iN})$. Authority A_j computes

$$\prod_i h_{ij} = \prod_i E_j(s_j(v_i)) = E_j(\sum_i s_j(v_i))$$

as follows from the homorphic property. A_j decrypts the sum of its shares $S_j = \sum_i s_j(v_i))$. S_j together with the proof of correct decryption is made public. Let A be the set of $t+1$ authorities that were successful in decrypting their shares S_j and that passed the proof of correct decryption. The final sum of the votes can be computed by anyone as

$$S = \sum_{j \in A} S_j \lambda_j = \sum_{j \in A} \left(\sum_i s_j(v_i) \right) \lambda_j = \sum_i \sum_{j \in A} s_j(v_i) \lambda_j = \sum_i v_i$$

where λ_j are Lagrange coefficients.

Properties and Extensions

Eligibility. Only eligible voters are allowed to vote, and at most once. The voter cannot cast invalid or double vote, as he is required to prove the validity of his vote.

Privacy. The voter's privacy is protected by the encryption scheme. Any set of at most t authorities can gain nothing about his vote.

Universal Verifiability. This scheme is universally verifiable: Anyone can verify the validity of the sent votes, anyone can multiply the encrypted shares per each authority and anyone can verify whether the authority correctly decrypted its sum of shares. From these sumn of shares anyone can compute the final result.

Receipt-Freeness. The voter's receipt consists of the random parameters $k_i j$ used in the encryption of his vote.

1-out-of-L voting. This scheme can be directly extended to 1-out-of-L voting scheme: permutation $v_1 \ldots v_L$ of L possibilities is encrypted to $h = (h_1 \ldots h_L)$. L possibilities can be represented as $1, M, M^2 \ldots M^L$ (M is the number of voters). This way, we can easily derive result of the elections from the sum of the votes.

Weighted-voting. Furthermore, extension to weighted-voting is also straightforward: if the voter's vote should be counted k times, his L-tuple will be $(kv_1 \ldots kv_L)$.

Communication Complexity. In 1-out-of-L voting the h is L-tuple and consists of $O(LN)$ group elements. To prove its validity, the voter has to sent T more L-tuples and reveal either their structure or their relation to the h. Overall, the voter has to transmit $O(TLN)$ group elements to the bulletin board.

5.3.2 Schoenmakers' Scheme

The scheme is from [Sch99]. Similarly to the Benaloh's scheme, the voter shares his vote among the authorities using secret sharing scheme. In this case it is publicly verifiable secret sharing described in the section 4.3.

Computing the final tally exploits the homomorphic property of the secret sharing (the tally – sum of the secrets is reconstructed from the multiplied shares).

Initialization stage. Initialization of the publicly verifiable secret sharing scheme is performed (generators g, G of Z_p, public keys $h_j = g^{z_j}$ of the authorities are published).

Voting stage. Voter V_i chooses his vote $v_i \in \{0, 1\}$ and a random $s_i \in Z_p$. He runs the distribution protocol (from the section 4.3) to share a secret g^{s_i} and publishes the value $U_i = g^{s_i+v_i}$. In addition, to show that indeed $v_i \in \{0, 1\}$ he gives a proof that

$$\log_G C_0 = \log_g U_i \ \vee \ \log_G(GC_0) = \log_g U_i$$

($C_0 = G^{s_i}$ is published as a part of the distribution protocol). A proof from the section 4.12.4 is suitable.

Anyone can check the ballot in the bulletin board due to the public verifiability of the secret sharing scheme and the given proof of $v_i \in \{0, 1\}$.

Counting stage. Suppose that the voters V_i, $i = 1, \ldots m$ succeeds in casting valid ballots. Firstly, all the respective encrypted shares are accumulated

$$H_j^* = \prod_i H_{ij} = \prod_i h_j^{p_i(j)} = h_j^{\sum_i p_i(j)}$$

Next, each authority A_j applies a reconstruction protocol from the section 4.3 to obtain $g^{\sum_i p_i(0)} = g^{\sum_i s_i}$, due to the homomorphic property. Combining with $\prod_i U_i = g^{\sum_i s_i+v_i}$ we gain $g^{\sum_i v_i} = g^T$. The final tally T can be now computed as in the CGS-scheme (see section 5.3.3).

Achieved Properties and Extensions

Eligibility. Only eligible voters can write into the bulletin board. The voter cannot cast invalid or double vote, since he is required to give a proof of the validity of the vote.

Privacy, Robustness. Any set of at most t authorities can tell nothing about the voter's vote. Privacy is protected by the security of the publicly verifiable secret sharing scheme.

Universal Verifiability. Anyone can verify the validity of the voter's vote. The publicly verifiable secret sharing scheme prevents the voter from distributing invalid shares to the authorities. Anyone can multiply the encrypted shares of the jth authority, and anyone can verify whether the jth authority had decrypted its sum of the shares correctly. Anyone can compute the final tally from the published sums of the shares.

Receipt-Freeness. The scheme is not receipt-free, since the voter's receipt is the secret s_i.

1-out-of-L Voting. We propose the following extension to 1-out-of-L voting:

The voter V_i chooses his vote v_i from the set $\{1, M, M^2, \dots M^{L-1}\}$, distribute the secret g^{s_i} among the authorities and publishes the value $U_i = g^{s_i+v_i}$. The proof of validity of the vote boils down to the proof of

$$\log_G(GC_0) = \log_g U_i \ \vee \ \log_G(G^M C_0) = \log_g U_i \ \dots \ \vee \log_G(G^{M^{L-1}} C_0) = \log_g U_i$$

(recall that $C_0 = G^{s_i}$ is published as a part of distribution protocol). Just use the non-interactive proof from the section 4.12.4.

The authorities decrypt the value $\sum v_i$, from which the result of the election can be easily computed.

Communication Complexity. Distribution of the secret s_i to the authorities requires to send $O(N)$ group elements to the bulletin board. The non-interactive proof of the validity of the vote in 1-out-of-L voting consists of $O(L)$ group elements. Overall, the voter has to write $O(N + L)$ group elements to the bulletin board.

5.3.3 CGS-Scheme

This schemes comes from [CGS97]. It is very efficient and satisfies all requirements except for the receipt-freeness. It is based on the discrete log assumption, and it can be modified for q-th residuosity assumption.

Yes/No Voting scheme

Voters publicly send their votes encrypted by ElGamal cryptosystem. The decryption key is shared between the authorities. At the end of the election, votes are multiplied and the authorities decrypt the sum of the votes as the result of the election.

Initialization stage. Robust threshold ElGamal cryptosystem is set up (described in the section 4.10); the authorities share the decryption key s. Public key (p, g, h), commitments of the shares $h_j = g^{s_j}$ and a fixed generator G of G_q are published.

Voting stage. The voter V_i chooses his vote: $m_0 = G$ for yes-vote, $m_1 = 1/G$ for no-vote. The encrypted vote is of the form $(x, y) = (g^k, h^k m_b)$, where k is random and $b \in \{0, 1\}$. Voter adds a proof that his vote is of the correct form. For this, a non-interactive proof that $\log_g x = \log_h(y/G) \vee \log_g x = \log_h(yG)$ is used. An interactive proof from the section 4.12.4 is suitable; it can be made non-interactive using the Fiat-Shamir technique described in the section 4.12.1. Ballot (encrypted vote + proof of validity) is sent to the bulletin board.

Counting stage. The proofs of validity are checked by the authorities and the product of all valid encrypted votes $(X, Y) = (\prod_i x_i, \prod_i y_i)$ is formed. Finally, the authorities jointly execute the decryption protocol from the section 4.10 for (X, Y) to obtain the value of $W = Y/X^s$. Each authority also publishes a non-interactive proof (from the decryption protocol) that it has really used its part of the shared key.

Thus, we get $W = G^T$, where T is a difference between the number of yes-votes and no-votes; $-M \leq T \leq M$; M is a number of eligible voters. Hence, $T = \log_G W$, which is in general hard to compute. The value of the T can be determined using $O(M)$ modular multiplications by iteratively computing $G^{-M}, G^{-M+1}, \cdots$ until W is found.

1-out-of-L Voting Scheme

There are numerous approaches to extent the previous yes/no voting scheme to 1-out-of-L elections. One is as follows:

We simply take L generators $G_1, \ldots, G_L$, and accumulate the votes for each option separately. The proof of validity of the ballot (x, y) now boils down to a proof of knowledge of

$$\log_g x = \log_h(y/G_1) \ \vee \cdots \vee \ \log_g x = \log_h(y/G_L)$$

The voter can generate this proof only for one generator G_j. Thus, it is automatically guaranteed that he will vote only for one option.

Problem arises in the computing the final tally. The decryption of the product of all valid votes leads to $W = G_1^{T_1} G_2^{T_2} \cdots G_L^{T_L}$. The values $T_1, \ldots, T_L$ can be computed using $O(M^{L-1})$ multiplications[3].

Achieved properties

Eligibility. Incorrect ballots of malicious voters will not pass through the proof of validity. The scheme is resistant up to t malicious authorities.

[3]Note that the condition $\sum_{i=1}^{L} T_i = m$, $m \leq M$ (m is the number of the voters participating in the election) can be exploited by reducing the problem to a search for $T_1, \ldots, T_{L-1}$ satisfying

$$W/G_L^m = (G_1/G_L)^{T_1} (G_2/G_L)^{T_2} \cdots (G_{L-1}/G_L)^{T_{L-1}}$$

The naive method (checking all possible combinations) needs time $O(m^{L-1})$, and it can be improved considerably by a generalization of the baby-step giant-step algorithm (see [DL01]) of time $O((\sqrt{m})^{L-1})$

Privacy. Privacy of the individual votes is guaranteed partly by the security of ElGamal cryptosystem. Individual vote is hidden for any set of at most t authorities.

Universal Verifiability. Any observer can check the proofs of validity of the ballots, any observer can make a product of the valid votes, and any observer can verify the correctness of the decryption by checking the proofs of authorities of using correct shares.

Receipt-Freeness. Voter can prove to any third party how he has voted just by showing randomness k used in the ElGamal encryption. Therefore, this scheme is not receipt-free.

Communication Complexity. In 1-out-of-L voting, voter's ballot consists of 2 group elements forming the encrypted vote and $2L+1$ group elements belonging to the non-interactive proof of the validity of the vote. Overall the voter needs to transmit $2L+3$ group elements to the bulletin board.

Further extensions and modifications

K-out-of-L voting. Recall that in this type of the voting the voter selects K from L possibilities, and the selected possibilities should be distinct.

As in the 1-out-of-L voting, L possibilities are represented by generators $G_1 \cdots G_L$. The final tally will be obtained from the form $G_1^{T_1} G_2^{T_2} \ldots G_L^{T_L}$, where $T_i \leq KM$.

The voter sends K encrypted votes $(x_1, y_1), \ldots, (x_K, y_K)$. He needs to prove that these ballots contain valid possibilities and that these possibilities are distinct.

Take a pair of the ballots $[(x_i, y_i), (x_j, y_j)]$, $i < j$. If

$$(x_i, y_i) = (g^{k_i}, h^{k_i} G_r)$$

is the encryption of the G_r and the

$$(x_j, y_j) = (g^{k_j}, h^{k_j} G_s)$$

is the encryption of the G_s, we say that the $[(x_i, y_i), (x_j, y_j)]$ is the encryption of $[G_r, G_s]$.

It is enough to prove for all pairs of the ballots $[(x_i, y_i), (x_j, y_j)]$, $i < j$ that this pair is the encryption of an element from the set

$$\{[G_r, G_s] \mid r, s = 1 \ldots L, r \neq s\}$$

This can be achieved as follows:

The product of the pair of ballots

$$(x_i x_j, y_i y_j) = \left(g^{k_I + k_j}, h^{k_i + k_j} G_r G_s\right)$$

is the encryption of G_rG_s. Hence, it is enough to prove that (x_ix_j, y_iy_j) is the encryption of the one element from the set $S = \{G_rG_s \mid r, s = 1, \ldots, L, r \neq s\}$. The set S contains $\frac{L(L-1)}{2}$ elements. Therefore, we use 1-out-of-$\frac{L(L-1)}{2}$ re-encryption proof from the section 4.12.3.

To sum up, the voter sends K encrypted votes to the bulletin board, which are $2K$ group elements. Furthermore, he is required to give $\frac{K(K-1)}{2}$ 1-out-of-$\frac{L(L-1)}{2}$ re-encryption proofs. Altogether, he transfers

$$2K + \frac{K(K-1)}{2}\left(2\frac{L(L-1)}{2} + 1\right) = O(K^2L^2)$$

group elements to the bulletin board.

1-L-K voting. We assume that each of the L sets contains n elements. ith element of the rth set is represented as $G^{(rn+i)M}$, where G is a fixed generator and M is the number of voters. Voter sends K different elements from $\{G, G^M, G^{2M}, \ldots, G^{nM}\}$ as in the K-out-of-n voting. Moreover, he has to specify from which set these elements should be chosen. He sends $B = (g^k, h^kG^{rnM})$ for the rth set. Each of the previously sent B_i is multiplied with B to get the desired $B_iB = (g^{k_i+k}, h^{k_i+k}G^{(rn+i)M})$.

To specify the set his favorites belongs to, the voter has to transmit $2+2L+1$ group elements (2 elements form an encryption of the set, $2L+1$ elements are needed for non-interactive proof). Selecting K elements from the set is in fact K-out-of-n voting that transmits $O(K^2n^2)$ elements. Altogether the number of transmitted group elements grows to $O(L + K^2n^2)$.

Alternative cryptosystem. We have modified the ElGamal cryptosystem to encrypt the message m as $E(m) = (g^k, h^kG^m)$. By this modification we gain the desired homomorphic property $E(m_1)E(m_2) = E(m_1 + m_2)$ instead of the ElGamal's homomorphism $E(m_1)E(m_2) = E(m_1m_2)$. However, our modified system does not present a trapdoor to decrypt the message m from the $E(m)$ directly; we need to assume that the message space is small enough to make the exhaustive search through all possible messages feasible.

This assumption is very strong, and it is not satisfied when the L, M are quite high. Consider for instance the 1-out-of-L voting with $M = 1000000$ eligible voters and $L = 100$ candidates; the decryption of the final tally would require about $\sqrt{1000000^{99}} \approx 10^{300}$ operations.

What we need is a public-key cryptosystem, threshold and homomorphic in the sense $E(m_1)E(m_2) = E(m_1 + m_2)$, allowing direct computation of the message m from $E(m)$, without exhaustive search. Possibility to prove that $E(m)$ is the encryption of m_1 or m_2 in a zero-knowledge manner is also important to ensure validity and correctness of the vote. Threshold version of the Paillier cryptosystem (see [FPS00], [DJ01]) is suitable.

Schoenmakers [Sch99] proposed a similar approach based on the publicly verifiable secret sharing.

Lee and Kim [LK00] modified this scheme to achieve receipt-freeness. An honest verifier re-encrypts the voter's vote, and together with the voter con-

structs a proof of its correctness. All communication between the voter and the honest verifier is done through the untappable channel. It is supposed that the "honest verifier" does not collaborate with the dishonest voter trying to cast invalid or double vote.

5.4 HS-Scheme

The scheme comes from [HS00].

The possibility of coercing in the CGS-scheme is caused by the randomness k used in the encryption of the vote. The problem is that when the voter himself encrypts his vote, he knows what and how is encrypted, and he can be coerced to reveal it. In the HS scheme, possible votes are encrypted and permuted by the authorities, one after another. A permutation of the encrypted votes is sent through the untappable channel to the voter. Voter just points at the vote he chooses. The scheme is designed in such a way that only the voter gets to know the final permutation and he can lie about it to anyone else.

5.4.1 1-out-of-L Voting Scheme

Votes will be encrypted as in the CGS scheme (section 5.3.3), using El-Gamal encryption. Recall that ith choice encrypts to the $(g^k \bmod p, h^k G_i \bmod p)$, where (p, g, h) is the public El-Gamal key and k is a random number.

Initialization stage. Similarly to the CGS scheme, N authorities set up robust threshold El-Gamal cryptosystem. The generators $G_1, G_2, \ldots G_L$ representing the possible choices are published. Authorities also create a public list of all standard-encrypted valid votes $e_1^{(0)}, e_2^{(0)}, \ldots, e_L^{(0)}$, where $e_i^{(0)}$ is the encryption of the ith choice G_i using the randomness $k = 0$: $e_i^{(0)} = (1, G_i)$

Voting stage. For each voter V, the authorities generate a list of the encryptions of all possible votes, from which the voter V selects the one representing his intention.

In turn, for each authority A_j (where $j = 1, 2, \ldots, N$):

Roughly said, the A_j takes on input the list $e_1^{(j-1)}, \ldots, e_L^{(j-1)}$, re-encrypts each element from the input list, and permutes the list in a random order. This way, A_j produces the output list $e_1^{(j)}, \ldots, e_L^{(j)}$. Moreover, A_j is required to prove that the output list has been properly constructed. If the A_j fails in some way or the voter objects to A_j, then the A_j is ignored and it is put $e^{(j)} = e^{(j-1)}$. The first authority picks the standard list $e_1^{(0)}, \ldots, e_L^{(0)}$ as the input.

Below is the more detailed description of the protocol:

1. A_j computes the output list $e_1^{(j)}, \ldots, e_L^{(j)}$ as follows:
   (a) A_j selects random permutation $\pi_j : \{1 \ldots L\} \to \{1 \ldots L\}$ and the random numbers $k_1, \ldots, k_L \in_R Z_p$.
   (b) The $\pi_j(i)$th item in the final list is obtained by re-encrypting the ith item $e_i^{(j-1)}$ from the input list with the randomness k_i.

2. Without revealing the permutation π_j as well as the randomness $k_1, \ldots, k_L$, the A_j shows that the output list $e_1^{(j)}, \ldots, e_L^{(j)}$ is correctly constructed:

   For each $i = 1 \ldots L$ the A_j proves that there exists a re-encryption of the ith item e_i^{j-1} of the input list in the output list $e_1^{(j)}, \ldots, e_L^{(j)}$. For this, the non-interactive version of the 1-out-of-L re-encryption proof from the section 4.12.3 is used.

3. A_j secretly transfers the permutation π_j together with the private proof of its correctness through the untappable channel to the voter V. More precisely, A_j proves that for each i, $e_{\pi_j(i)}^{(j)}$ is the re-encryption of $e_i^{(j-1)}$. Designated-verifier re-encryption proof from the section 4.12.5 is suitable.

4. If the voter does not accept the proof, he publicly complains about the authority. After that, the list is rolled back to the previous state $e_1^{(j-1)}, \ldots, e_L^{(j-1)}$ and the jth authority is ignored. The voter may complain against at most $N - t - 1$ authorities.

The voter publicly announces the position i of his desired vote in the final list $e^{(N)}$.

Counting stage. Each voter has already chosen his vote from the final list produced for him by the authorities. His vote is encrypted in the same way as in the 1-out-of-L voting in CGS scheme. Hence, the result of the election can be computed in the same way as in the CGS scheme (see section 5.3.3) – the encrypted votes are multiplied to obtain the encrypted sum of the votes, the authorities jointly decrypt the sum of the votes and publish the proof of correct decryption.

Achieved Properties

If the El-Gamal cryptosystem is secure, then this scheme provides eligibility, universal verifiability, computational privacy, robustness and receipt-freeness.

Eligibility. Eligibility is achieved, as the voter cannot cast invalid vote, he votes at most once and he votes how he wishes (he can trace the permutation of the encrypted votes). Only one problem arises, when one of the authorities coerces. It can force the voter not to complain against its wrong proof of the permutation. Therefore, the voter will lose his track and he will vote randomly. For the coercer, random vote can be better than the almost sure vote for his enemy.

Privacy, robustness. The encrypted vote cannot be decrypted by an outsider or by a group consisting of at most t authorities. Given a list of encrypted votes and a shuffled list, it is infeasible to find out the correct permutation. Voter can skip at most $N - t - 1$ shufflings, so at least $t + 1$ shufflings were

performed correctly. The colluding set of at most t authorities cannot find out the reordering of the list.

Universal Verifiability. Any observer can check whether the authority A_j shuffled the list correctly. This way, any observer can verify whether the last list $e^{(N)}$ is the list of the encryptions of all possible votes. Any observer can compute a product of all encrypted votes selected by the voters, and any observer can verify whether the final tally has been correctly decrypted by the authorities.

Receipt-freeness. Assume that the coercer or vote-buyer does not collude with the authorities. Voter can interact only at two points: he can revert shuffling of at most $N - t - 1$ authorities, and he points at the encrypted vote of his choice. From each authority he receives permutation together with the proof of its correctness. Untappable channel guarantees that no one is able to intercept this message. Thus, the voter can substitute received data with his own permutation, and due to the no transferability of the designated-verifier proof, he can construct a proof of its correctness as well. This way he can deceive the possible coercer.

If the coercer colludes with some of the authorities (at most with t of them), he will know the permutations they made. Therefore, the voter cannot lie about their permutations. If he knows the colluding authorities, he will lie about the permutation of reliable authority. Otherwise he selects one authority randomly and lies for its permutation. Apparently, receipt-freeness holds as long as the voter knows at least one authority not colluding with the coercer. Coercer can only force the voter to vote randomly.

Communication Complexity. Per each voter, in the bulletin board should be stored N lists each containing L items ($2NL$ group elements), NL 1-out-of-L re-encryption proofs ($NL(2L+1)$ group elements), the voter's complaints and the index i of the voter's vote in the final list. Overall, $O(NL^2)$ group elements have to be transferred and stored in the bulletin board.

The voter receives N permutations and NL designated-verifier re-encryption proofs through the untappable channel ($NL \log_2 L$ bits and $4NL$ group elements).

Extensions

K-out-of-L voting. Simply, voter points at K votes from the final list.

1-L-K voting. Similarly to CGS scheme (see section), ith element from the rth set is represented as $G^{(rn+i)M}$. Authorities shuffles two lists: first stands for selecting a set $\{1, G^{nM}, G^{2nM}, \dots G^{(L-1)nM}\}$; the second for selecting K possibilities $G, G^M, G^{2M}, \dots, G^{nM}$. Finally, the voter points at K elements from the second list and to the one element B from the first list. In order to

get the proper votes, each of the K elements selected from the second list is multiplied with the B.

6 Implementation of the Untappable Channel – Deniable Encryption

Existence of the untappable channel is sometimes assumed between the voter and the authority. The sent messages cannot be intercepted by anyone, thus nothing is revealed to public and the voter may claim he has sent or received any message he wants without being found lying.

Untappable channel can be realized by providing multiple channels between the sender and the receiver under the assumption that the coercer cannot tap on all channels simultaneously. The sender sends the message through one of the channels randomly selected. If the coercer can tap merely on a few channels, he can intercept the message only with small probability.

If this approach is not applicable, we have to assume that tapping on the channel between the voter and the authority is likely. Nevertheless, intercepting the communication between the voter and the authority should be useless for the coercer; the voter should be able to mislead him by interpreting the intercepted communication in the coercer's desired way. The coercer should not obtain any information about the transmitted message, except for its length.

Standard cryptographic implementations of the secure channel are not suitable. If the adversary intercepts the transmission of the encrypted message and later forces the sender to reveal the secret keys and the random choices generated during the encryption, the cleartext is exposed.

Using traditional encryption methods the sender is bounded to the cleartext. For an intercepted ciphertext, the sender cannot generate fake secret keys and fake random choices that will persuade the adversary that the given ciphertext corresponds to the different cleartext. For instance, ElGamal encryption $(x, y) = (g^k, h^k m)$ of the message m can be opened by revealing the random choice k. The sender is not able to generate any k' opening the ciphertext as the encryption of a different message m'.

This may seem rather confusing – if the ciphertext c can be interpreted as the encryption of the message m as well as the encryption of the message m', how can the receiver distinguish between these cases and decrypt the message in the sender's intentioned way? The receiver has to possess some secret information unavailable to anyone (maybe except for the sender). With this trapdoor, he is able to decrypt the message correctly. Without it, the coercer cannot deduce whether the given interpretation of the ciphertext is correct.

An encryption scheme allowing the sender generating his fake random choices and his fake secret keys that will make the ciphertext looks like the encryption of a different cleartext is called *sender-deniable encryption*. An encryption scheme satisfying analogous requirements for the receiver is called *receiver-deniable encryption*. The concept of deniable encryption was introduced in [CDNO].

One-time pad is an example of the deniable encryption. The sender and the receiver shares a secret key s. A message m is encrypted to $c = s \oplus m$. When an adversary attacks the sender (receiver), the sender (receiver) may claim to share a key $s' = c \oplus m'$ instead of s to convinced him of sending (receiving) the message m'. However, the key s may be used only once.

We will focus on the sender-deniable encryption, as the receiver-deniable encryption can be constructed from the sender-deniable encryption. Let the sender S wants to transmit a message m of the length l to the receiver R. S and R execute the following protocol:

1. R generates a random message r of the length l
2. R uses sender-deniable encryption to transmit the message r to S
3. S sends $c = m \oplus r$ through the public channel to R
4. R extracts the message m as $m = c \oplus r$

When the receiver is under attack, he may claim to send different message r and thereby receiving a different message m.

6.1 Deniable Encryption Based on Quadratic Residues

We propose the following sender-deniable public-key encryption scheme. The generalization of this scheme and more constructions of the sender-deniable encryption schemes (from [CDNO]) can be found in the next section 6.2. The sender encrypts the message using the public-key of the receiver and he can later fake his random choices. The message is encrypted per bits.

The receiver has RSA modulus $n = pq$, where p, q are large prime numbers. n is his public key, and p, q are kept secret. The sender has no public or secret key.

Some basic facts about the quadratic residues and pseudo-squares can be found in the section 4.5. The set of quadratic residues and pseudo-squares will be denoted by J_n.

Encryption. A sender transmits one bit to the receiver as follows (k is security parameter):

- If the sender wants to transmit 1, he sends k random quadratic residues: he chooses $x_i \in_R Z_n^*$, $i = 1 \dots k$ at random and sends $y_i = x_i^2 \bmod n$ to the receiver.
- If the sender wants to transmit 0, he sends k random integers $y_i \in J_n$, $i = 1 \dots k$: he selects $y_i \in_R Z_n$ and checks whether $\left(\frac{y_i}{n}\right) = 1$.

Decryption. The receiver decrypts the message as follows:

- If all k received y_i are quadratic residues, he decodes the message as a 1.
- If at least one of the k received y_i is not a quadratic residue, he decodes the message as a 0.

As only the receiver can distinguish between the quadratic residues and quadratic non-residues, only he can decrypt the message.

If the sender transmits a 1, then all y_i are quadratic residues and the receiver decodes the message correctly. If the transmitted message is 0, there is a (small)

probability that all k randomly selected y_i are quadratic residues. Consequently, the receiver will decrypt the message incorrectly as a 1. The probability that all k randomly selected $y_i \in J_n$ are quadratic residues is $(1/2)^k$, which can be considered negligible.

The sender may interpret the sent ciphertext to the adversary in the following ways:

- If the sent bit was 1, he can open the ciphertext as a 1 by revealing the square roots x_i, or as a 0 by claiming that all y_i has been chosen randomly.
- If the sent bit is 0, he can open the ciphertext only as a 0.

As the adversary cannot distinguish between the quadratic residues and non-residues, it has to accept the sender's claim that the sent y_i has been chosen randomly, even if they were chosen from the quadratic residues.

The sender cannot cheat if the transmitted bit is 0. We can overcome this problem by extending the encryption scheme:

1. The sender picks a random bitstring s of the length l:
   - In order to transmit 0, he chooses s with even number of 1's
   - In order to transmit 1, he chooses s with odd number of 1's
2. The sender transmits s per bits in the manner described above.

So, the number of transmitted elements grows from k to lk. If the receiver receives s with odd number of 1's, he decodes it as a 1. Otherwise, he decodes the message as a 0.

When the sender is under attack, he may declare one 1 from the bitstring s to be a 0. This way, the parity of the number of 1's in the s has changed and the bit encoded in the s is inverted. Cheating is impossible only in the case when the s does not contain 1 ($s = 0^l$).

6.2 The Generalized Parity Scheme

The idea presented in the previous section is not restricted to the use of quadratic residues and can be generalized. In fact, this method can be applied to any subset $S \subset X$ of the set X with the following properties:

1. S is not too large: $|S| \leq \frac{|X|}{2}$.
2. Given $x \in X$ and a secret (trapdoor) d, it is easy to determine whether $x \in S$.
3. It is easy to generate a random element $x \in S$, even without the secret d.
4. Without d, it is infeasible to distinguish between the values uniformly chosen from the S and the values uniformly chosen from the X.

Let's denote the number of elements in the S as $m = |S|$ and the number of elements in the X as $n = |X|$. In the previous encryption scheme, X is the set of all pseudo-squares and quadratic residues, the set S is the set of all quadratic residues modulo n and the trapdoor d is the factorization of the modulus n. The encryption, decryption and the faking algorithms look as follows:

Encryption. Again, to send a bit 1 (0) a random string of bits $s = s_1 \cdots s_l$ such that the number of 1's in s is odd (even) is selected. Further, for $i = 1, 2, \ldots, l$ s_i is encrypted to r random elements from the S if $s_i = 1$, or to r random elements from the X if $s_i = 0$. The probability of the incorrect decryption of the s_i is $\left(\frac{m}{n}\right)^r$ and should be at most $\frac{1}{2^k}$, so we choose r as the smallest integer satisfying $\left(\frac{m}{n}\right)^r \leq \frac{1}{2^k}$.

Decryption. Firstly, the string of bits s is decrypted per bits. Exploiting the secret d, it is easy to decide whether all of the r elements making up the encryption of the s_i are elements of the set S. In that case, s_i is decrypted as a 1, otherwise as a 0. The sent bit is obtained from the parity of the number of 1's among the s_i.

The probability of the incorrect decryption is less than $\frac{l}{2^k}$ ($\frac{1}{2^k}$ is the upper-bound of the probability that the s_i is decrypted erroneously).

Faking Algorithm. If $s = 0^l$, faking is impossible. Otherwise, the sender picks up one $s_i = 1$ and claims that $s_i = 0$ (instead of revealing how the encryption of the s_i – r elements from the S – has been created, he claims that these r elements has been chosen randomly from the X).

R. Canetti, C. Dwork, M. Naor and R. Ostrovsky in [CDNO] proposed the following constructions of the sets S, X described above. Both constructions assume the existence of one-way trapdoor one-to-one function $f : W \to W$ (computing f^{-1} is feasible only with the secret d) and a predicate $B : W \to \{0, 1\}$ satisfying

$$|\{w \in W |\ B(w) = 1\}| = \frac{|W|}{2}$$

Computing $B(w)$ is easy for all $w \in W$.

Construction A. The sets X, S are defined as:

$$X = W$$

$$S = \{w \in W |\ B(f^{-1}(w)) = 1\}$$

It is easy to see that $|S| = \frac{|X|}{2}$. A random element $y \in S$ can be obtained by choosing a random $x \in W$ such that $B(x) = 1$ and computing $y = f(x)$. With the trapdoor d, it is easy to determine whether the given $y \in X$ is also a member of S; just check whether $B(f^{-1}(y)) = 1$. Without the trapdoor d, given $y_1, \ldots, y_k \in X$ the coercer cannot determine whether these values have been chosen randomly from the S or randomly from the X: The coercer cannot compute f^{-1}; therefore he has no information about the values $x_i = f^{-1}(y_i)$ and no information about $B(x_i)$.

Construction B. Another technique (which turns to be more efficient) for obtaining the set S is as follows.

$$X = \{wb|\ w \in W, b = b_1 \dots b_k; b_i \in \{0,1\}; i = 1 \dots k\}$$

$$S = \{wb|\ wb \in X; B(f^{-i}(w)) = b_i; i = 1 \dots k\}$$

Notice that for any $w \in W$ there exists exactly one b such that $wb \in S$; and this $b = b_1 \dots b_k$ where $b_i = B(f^{-i}(w))$. Hence

$$|S| = \frac{|X|}{2^k}$$

An element $wb \in S$ can be randomly generated by choosing $x \in W$ at random and setting $w = f^k(x)$, $b_i = B(f^{k-i}(x))$.

Determining whether the given $wb \in X$ is also a member of S is easy with the knowledge of the trapdoor d: just compute $f^{-1}(w), f^{-2}(w), \dots, f^{-k}(w)$ and check whether $B(f^{-i}(w)) = b_i$ for $i = 1 \dots k$.

Without the trapdoor d, given $wb \in X$ the coercer cannot determine whether this value has been chosen randomly from the S or randomly from the X: Since the coercer cannot compute f^{-1}, he has no information about the values $x_i = f^{-i}(w)$, no information about $B(x_i)$ and therefore he cannot decide whether $b_i = B(x_i)$ for all i.

Communication Complexity. In order to transmit 1 bit through untappable channel, the construction A transfers kl elements of the W. Conversely, the construction B transmits only l elements of the W and kl bits.

7 The Proposed Scheme

The HS-scheme seems to be efficient and practicable. However, the amount of work done by the voter and the number of bits sent through the untappable channel are quite high. The voter needs to check LN proofs sent to him through the untappable channel in order to keep track of the permutation of the votes. We would prefer the voter's computation and the amount of data transmitted through the untappable channel to be independent on L.

7.1 Requirements for the Voting Protocol

Similar to the HS-scheme, the votes will be encrypted by the robust threshold public-key cryptosystem with the homomorphic property. The HS-scheme required this public-key cryptosystem to support random re-encryptability, 1-out-of-L re-encryption proof and designated-verifier re-encryption proof. Our scheme, in addition, requires the existence of 0-preserving re-encryption with the 0-preserving re-encryption proof (see section 7.1.1) and 1-out-L 0-preserving re-encryption proof (see section 7.1.2).

The modified ElGamal cryptosystem $E(m) = (g^k, h^k G^m)$ (section 5.3.3) satisfies all mentioned requirements. Its only disadvantage is relatively small message space caused by inefficient decrypting algorithm (problem of computing m from G^m).

Without loss of generality, assume that there exists a standard encryption $e^0 = E^0(m)$ of the message m. In the modified ElGamal cryptosystem put $E^0(m) = (1, G^m)$.

7.1.1 0-preserving re-encryption

0-*preserving re-encryption* is an algorithm that on the input $e = E(m)$ outputs $e' = E(m')$, where $m' = 0$ if and only if $m = 0$, otherwise $m' \neq 0$ is a random message (uniformly distributed).

In addition, a proof that e' is a 0-preserving re-encryption of e, not revealing the relation between the messages m and m', is required.

In the modified ElGamal cryptosystem ($E(m) = (g^k, h^k G^m)$), the 0-preserving re-encryption looks as follows: Given $E(m) = (x, y)$, $m \in Z^*_{p-1}$, compute the 0-preserving re-encryption $E(m') = (x', y')$ as

$$(x', y') = (x^r, y^r)$$

where r is random $r \in_R Z^*_{p-1}$.

If $(x, y) = (g^k, h^k G^m)$, then

$$(x', y') = (x, y)^r = (g^k, h^k G^m)^r = (g^{kr}, h^{kr} G^{mr}) = E(mr)$$

and $m' = mr$.

To prove that (x', y') is the 0-preserving re-encryption of the (x, y), it is enough to show that

$$\log_x x' = \log_y y'$$

The non-interactive proof from the section 4.12.2 is suitable. The value $r = \frac{m'}{m}$ should be kept secret.

7.1.2 1-out-of-L 0-Preserving Re-Encryption Proof

The prover wants to prove that for the encrypted message $e = E(m) = (x, y)$ there exists a 0-preserving re-encryption in the L encrypted messages $e_1 = (x_1, y_1), \ldots, e_L = (x_L, y_L)$.

Assume that the (x_t, y_t) is the 0-preserving re-encryption of (x, y) and that the re-encryption randomness is r, e.g. $(x_t, y_t) = (x^r, y^r)$.

Just consider the values $(x_1, y_1), \ldots, (x_L, y_L)$ to be the encrypted messages in the ElGamal cryptosystem (p, x, y) (instead of the original (p, g, h)). In this cryptosystem, the $(x_t, y_t) = (x^r, y^r)$ is the re-encryption of $(1, 1)$. Therefore, it is enough to prove that in the elements $(x_1, y_1), \ldots, (x_L, y_L)$ is the re-encryption of $(1, 1)$. The non-interactive proof from the section 4.12.3 is suitable.

7.1.3 1-out-of-L Encryption Proof

Given an encryption $e = E(m)$ of the message m, the authorities want to verify whether m is valid without revealing any other information about the m. The message m is valid if it is in the set of valid messages $\{m_1, m_2, \ldots m_L\}$. Note that the authorities can decrypt any message, but decrypting the m straightforwardly is not desired, as it reveals the m. We assume that at least $t + 1$ authorities will follow the protocol described bellow.

Assume that the $e_1^0, \ldots, e_L^0$ are standard encryptions of the valid messages $m_1, \ldots, m_L$. In the modified ElGamal cryptosystem, $e_i^0 = (1, G^{m_i})$.

1. The authorities compute the initial list $e_1^{(0)}, \ldots, e_L^{(0)}$ by setting

$$e_i^{(0)} = \frac{e}{e_i^0} = \frac{(g^k, h^k G^m)}{(1, G^{m_i})} = (g^k, h^k G^{m-m_i})$$

   Notice that $e_i^{(0)}$ is the encryption of 0 if and only if $m_i = m$. The message m is valid if and only if the initial list $e_1^{(0)}, \ldots, e_L^{(0)}$ contains the encryption of 0.

2. The authorities shuffles the initial list $e_1^{(0)}, \ldots, e_L^{(0)}$ to produce the final list $e_1^{(N)}, \ldots, e_L^{(N)}$ in the following manner. One by one, the authority A_j, $j = 1 \ldots N$ should do the following:

   A_j picks the list $e_1^{(j-1)}, \ldots, e_L^{(j-1)}$ on input and outputs the list $e_1^{(j)}, \ldots, e_L^{(j)}$. If the A_j fails in some way, then the A_j is ignored and it is put $e^{(j)} = e^{(j-1)}$. A_j generates the output list as follows:

   (a) A_j selects a random permutation $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$ and random numbers $r_1, \ldots, r_L$. A_j keeps π, $r_1, \ldots, r_L$ secret.

   (b) The $\pi(i)$-th item in the output list is the 0-preserving re-encryption of the i-th item from the input list using the randomness r_i.

   (c) For each item $e_i^{(j-1)}$ from the input list, the A_j proves that there is a 0-preserving re-encryption in the output list $e_1^{(j)}, \ldots, e_L^{(j)}$ using 1-out-of-L 0-preserving re-encryption from the section 7.1.2.

3. The authorities decrypt each element from the final list $e_1^{(N)}, \ldots, e_L^{(N)}$. The $e = E(m)$ encrypts valid message if and only if one of the decrypted values is 0.

The initial list contains the encryption of 0 if and only if the message m encrypted in $e = E(m)$ is valid. The authority A_j's output list contains the encryption of 0 if and only if the A_j's input list contains the encryption of 0, since each item from the output list is the 0-preserving re-encryption of one item of the input list. Therefore, the encryption e of the message m is valid if and only if the final list contains the encryption of 0.

The decrypted values tells nothing about the message m (except for its validity). The non-zero decrypted values look random, and their relations to the m are hidden.

In the last step, it is not necessary to actually decrypt the elements from the final list – the zero-test is sufficient. In the modified ElGamal cryptosystem, the authorities test whether $(g^k, h^k G^M)$ is the encryption of 0 as follows: they cooperate in obtaining G^M from $(g^k, h^k G^M)$ and test whether $G^M \stackrel{?}{=} 1$ (computing M from G^M is infeasible).

7.2 Introducing the Scheme

Scheme overview. In the proposed scheme, the voter shares his vote using $(t+1, N)$ secret sharing among the authorities. He sends the shares through the untappable channel. The authority publishes the re-encryption of the received share and sends to the voter designated-verifier proof of its correction. The voter may object to it. Those $t+1$ published re-encryptions against which the voter does not object are combined to produce the encrypted vote. Further, the authorities cooperate in ensuring its validity. All valid votes are multiplied and the authorities decrypt the sum of the votes.

In the 1-out-of-L voting, the voter V_i chooses his vote v_i from the set $\{1, M, M^2, \ldots M^{L-1}\}$ (M is the number of eligible voters). However, in the yes/no voting it suffices to encode the votes to $\{-1, 1\}$.

Initialization stage. Authorities set up the robust threshold public-key cryptosystem satisfying the mentioned requirements (modified ElGamal for instance).

Voting Stage. The voter V_i shares his vote v_i among the authorities using Shamir $(t+1, N)$ secret sharing scheme. For the ease of understanding, the communication between the voter and authorities is depicted in the figure 7.

1. The voter V_i generates a random polynomial f of degree t satisfying $f(0) = v_i$:
$$f(x) = v_i + a_1 x + \cdots + a_t x^t$$
The jth authority's share is $s_j = f(j)$. Recall that the vote v_i can be obtained from any set A of $t+1$ shares by computing
$$v_i = \sum_{j \in A} s_j \lambda_j$$

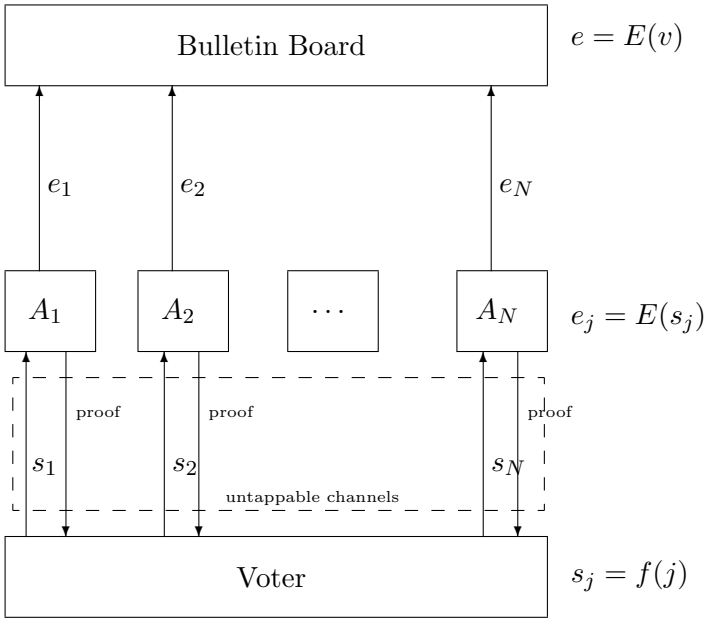

Figure 7: Communication in the scheme

where λ_j is the Lagrange coefficient.

2. The voter sends s_j through the untappable channel to the jth authority A_j $(j = 1, \ldots, N)$.

3. The authority A_j computes the standard encryption $e_j^0 = E^0(s_j)$, re-encrypts e_j^0 to e_j and writes e_j to the bulletin board. A_j keeps s_j, e_j^0 secret. Besides, A_j convinces the voter that e_j is the encryption of s_j. More precisely, A_j transmits a designated-verifier proof that e_j is the re-encryption of e_j^0 through the untappable channel to the voter.

4. V_i checks the received (for him designated) proofs. He may publicly complain about the incorrect ones; he can do so at most $N - t - 1$ times. In addition, V_i may point out the $t + 1$ authorities in which he trust.

5. Let A be the set of $t + 1$ authorities which the voter found trustworthy and against which he did not complain. The encrypted shares of these authorities are combined:
$$e = \prod_{j \in A} e_j^{\lambda_j} = \prod_{j \in A} E(s_j)^{\lambda_j} = \prod_{j \in A} E(s_j \lambda_j) = E(\sum_{j \in A} s_j \lambda_j) = E(v_i)$$
We see that the obtained e is the encryption of the vote v_i.

6. Furthermore, authorities verify the validity of e by executing the 1-out-L-encryption proof from the previous section 7.1.3. This proof is written to the bulletin board.

Counting stage. All valid votes e are multiplied, and the authorities jointly decrypt the sum of the votes.

Security Properties

Eligibility. If the voter V_i is honest and the e_j published by $t+1$ authorities he has selected as trustable are really the encryptions of the corresponding shares s_j, then the e constructed in the step 5 is obviously the encryption of the voter's vote v_i and the voter's vote is counted properly.

The dishonest voter V_i might either share an invalid vote, or distribute invalid shares (something that is not produced by secret sharing at all), or he might select the authority A_j as trustable despite the fact that the designated-verifier proof provided by A_j was incorrect. At any rate, the probability that the reconstructed vote e is valid is negligible. The invalidity of the e will be detected in the 1-out-of-L encryption proof and this invalid vote will not be counted.

Privacy. The voter's intention can be reconstructed from the $t+1$ shares. As the shares are sent through the untappable channel, they cannot be intercepted by anyone. The authorities publish the encryptions of their shares, so nothing can be deduced about them as far as the underlying cryptosystem is secure. The 1-out-of-L encryption proof verifying the validity of the vote also reveals no other information about it.

Verifiability. The scheme is universally verifiable. Anyone can read the encrypted shares e_j, the set A of $t+1$ authorities denoted by the voter as reliable, and he can construct the encrypted vote e. Furthermore, he can verify its validity by verifying the 1-out-of-L encryption proof. He can multiply all valid encrypted votes to obtain the encryption of the sum of the votes. Since the decryption is verifiable, he can also check whether the sum of the votes has been correctly decrypted.

Receipt-Freeness. This scheme is receipt-free if the voter knows the coercing authorities.

The voter can easily adjust the shares and the designated-verifier proofs according to his needs. If some authorities (at most t) collaborate with the coercer, the coercer gets to know their shares. If the voter knows which authorities are dishonest, he can still adjust the shares of the remaining $N-t$ honest authorities to fit the coercer's vote.

If the voter does not know the fraudulent authorities, he has to choose $N-t$ authorities randomly and he lies about their shares. The probability that he will guess the honest authorities is $\frac{1}{\binom{N}{t}}$, which is rather small. In the HS-scheme the voter has to guess only one honest authority, not all of them. The scheme has to be modified in order to satisfy this requirement. More precisely, the used secret sharing scheme has to be replaced with something else. In the $(t+1, N)$

secret sharing scheme, the $t+1$ shares commits the voter to the remaining $N-t-1$ shares. The voter can adjust only $t+1$ shares, since they determine the rest $N-t-1$ shares.

If the coercer computes s_j for the voter and the voter has no idea which t authorities are coercing, he can still do this simple (but risky) trick. He chooses $t+1$ "reliable" authorities, and from them "the most reliable one" (randomly, if he cannot distinguish between the honest and dishonest ones). Each authority A_j (except for "the most reliable one") will get the coercer's share s_j. He adjust the share of "the most reliable" authority such that the interpolation of the $t+1$ shares of "reliable" authorities gives his vote v_i. Of course, this distribution of the shares is incorrect, but as far as the "reliable" authorities sent back to him the correct designated verifier proof, his vote is counted properly. In the case that one "reliable" authority encrypts its share wrongly and sends to the voter wrong proof, the voter is in trouble. Either he will cast an invalid vote (by denoting the "reliable" authorities as trustable), or he will cast the coercer's vote (by denoting the other set not containing "the most reliable authority" as trustable).

7.2.1 The Modification Enhancing Incoercibility

Voting stage. For the ease of understanding, the communication in the voting stage is depicted in the figure 8.

1. The authority A_j chooses a random message s_j and encrypts it to $e_j = E(s_j)$. A_j keeps the s_j secret and writes e_j to the bulletin board. Furthermore, A_j sends s_j as well as a designated-verifier proof that e_j is the re-encryption of the $e_j^0 = E^0(s_j)$ to the voter through the untappable channel.

2. The voter V_i receives the s_j, and checks the designated-verifier proofs. He chooses at least $t+1$ authorities whose designated-verifier proofs are correct and in which he trusts. Let A be the set of the selected authorities. The voter computes $e_v = E(v_i - \sum_{j \in A} s_j)$ and writes A, e_v to the bulletin board.

3. The encrypted vote can be obtained by computing
$$e = e_v \prod_{j \in A} e_j = E(v_i - \sum_{j \in A} s_j) \prod_{j \in A} E(s_j) = E(v_i - \sum_{j \in A} s_j + \sum_{j \in A} s_j) = E(v_i)$$

4. The authorities verify the validity of e by executing the 1-out-of-L encryption proof.

Security Properties

Eligibility. The vote of the honest voter will always be properly counted. A dishonest voter might cast an invalid e_v. Consequently, the constructed e will be invalid with high probability, and this invalid vote will be rejected.

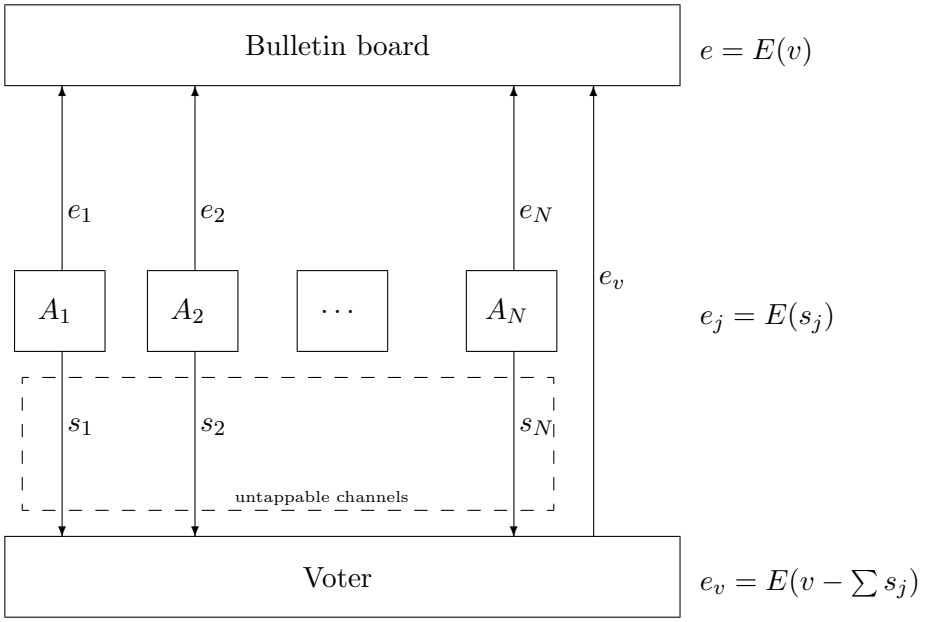

Figure 8: Communication in the voting stage

Privacy. Privacy is protected by the security of the underlying cryptosystem. The 1-out-of-L encryption proof reveals nothing about the voter's vote.

Incoercibility. The voter can adjust s_j according his needs. The values s_j, $j \notin A$ are not important. The published e_v commits him to the value $z = v_i - \sum_{j \in A} s_j$. Let the coercer's vote be v_c. The voter chooses one $k \in A$ and claims s_k to be $s_k = v_i - v_c + s_j$. He leaves the others s_j, $j \neq k$ unchanged.

8 Conclusion

We have summarized the existing voting schemes and their properties. We also proposed an efficient receipt-free voting scheme. This way, the goals of our work have been achieved.

The presented voting schemes have different security properties. The HS-scheme (section 5.4) is quite efficient and achieves privacy, verifiability and incoercibility. Untappable channel is used in the communication between the voter and the authority to prevent coercing. The amount of data sent through the untappable channel depends on the number of possible choices/votes (L).

However, known implementation of the untappable channel (section 6) is very inefficient (e.g. sending 1 bit through the untappable channel means sending 10^5 bits through the public channel). Therefore, we tried to cut down the amount of data that has to be sent through the untappable channel.

We succeeded in designing a scheme having the same security properties as the HS-scheme, but the amount of data sent through the untappable channel is reduced and does not depend on L (number of possible choices/votes).

The untappable channel (or deniable encryption) is the only way how to prevent coercing. Hence, in the scheme achieving incoercibility some information simply has to be sent through the untappable channel, and the question is how much it can be reduced.

Our scheme does not remove the dependence on L (number of possible choices/votes) completely, just from the voter's point of view. The computation done by the voter does not depend on L. Nevertheless, the verification of the validity of the vote performed by the authorities still depends on L.

Further work can be done in designing more efficient protocol ensuring the validity of the vote. The protocol can involve the authorities as well as the voter, should preserve incoercibility and its communication/computation complexity should be independent on L.

Our research has been focused on the schemes exploiting homomorphic property of the encryption method. Other kinds of schemes – schemes based on anonymous channel and blind signatures – can still be enhanced in security (incoercibility, making an objection, universal verifiability, etc.)

Bringing electronic voting into practice means to solve a lot of technical problems which have not been considered in this paper, especially authentication of the voter and secure platform problem. Current generation of personal computers is not sufficiently secure to act as the voting agent. Electronic voting is not like e-commerce, where the customer obtains a receipt confirming the transaction, or the parties involved in the transaction are recorded.

References

[Ben87] Josh Cohen Benaloh. *Verifiable Secret Ballot Elections*. PhD thesis, Yale University, 1987.

[Boy] Colin Boyd. A new multiple key cipher and an improved voting scheme.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. receipt-free secret-ballot elections (extended abstract). *Proc. 26th ACM Symposium on the Theory of Computing (STOCK)*, 1994.

[BY86] Josh Cohen Benaloh and Moti Yung. Distributing the power of the governement to enhance the privacy of voters (extended abstract). 1986.

[CDNO] Ran Canetti, Cynthia Dwork, Moni Naor, and Rafail Ostrovsky. Deniable encryption.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *Advances in Cryptology - EUROCRYPT*, 1997.

[Cha81] David L. Chaum. Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM 24*, Feb 1981.

[Cha88] David L. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa. *EUROCRYPT'88*, 1988.

[DJ01] Ivan Damgard and Mads Jurik. A generalization, a simplification and some applications of paillier's probabilistic public-key cryptosystem. *BRICS*, 2001.

[DK00] Ivan B. Damgard and Maciej Koprowski. Practical threshold rsa signatures without a trusted dealer. *BRICS RS-00-30*, November 2000.

[DL01] Vic DeLorenzo and Yiosiang Liow. The mathematics of cryptography: Discrete logarithm and elliptic curves. 2001.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. *Advaced in Cryptology - AUSCRYPT'92*, 1992.

[FPS00] Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. Sharing decryption in the context of voting or lotteries. *Financial Cryptography 2000*, LNCS, Springer-Verlag, 2000.

[GJKR99] Rosario Gennaro, Stanislav Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *EUROCRYPT'99*, 1999.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. *EUROCRYPT 2000*, 2000.

[Ive91] Kenneth R. Iversen. A cryptographic scheme for computerized general elections. *Advances in Cryptology - CRYPTO'91*, 1991.

[JL97] Wen-Sheng Juang and Chin-Laung Lei. A secure and practical electronic voting scheme for real world environment. *IEICE Trans. on Fundamentals*, E80-A(1), January 1997.

[JL99] Wen-Sheng Juang and Chin-Laung Lei. Partially blind threshold signatures based on discrete logarithm. *Computer Communications 22 (1999)*, pages 73–86, 1999.

[JLY98] Wen-Sheng Juang, Chin-Laung Lei, and Pei-Ling Yu. A verifiable multi-authorities secret elections allowing abstaining from voting. *International Computer Symposium, Tainan, Taiwan*, 1998.

[LK00] Byoungcheon Lee and Kwangjo Kim. Receipt-free electronic voting through collaboration of voter and honest verifier. 2000.

[MvOV96] A. Menezes, P. van Oorshot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting scheme for large scale election. *Proc. of Workshop on Security Protocols'97*, LNCS(1361), 1997.

[PIK93] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. *Advances in Cryptology - EUROCRYPT'93*, Springer-Verlag:248–259, 1993.

[Rad95] Michael J. Radwin. An untraceable, universally verifiable voting scheme. 1995.

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. *Advances in Cryptology - CRYPTO*, 1666 of Lecture Notes in Computer Science:148–164, 1999.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[SK94] Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. *Advances in Cryptology - CRYPTO'94*, Springer-Verlag:411–424, 1994.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme – a practical solution to the implementation of a voting booth. *Advances in Cryptology - EUROCRYPT'95*, 1995.