

# Coercion-Free Verifiable Internet Voting

Mirosław Kutylowski and Filip Zagórski

Institute of Mathematics and Computer Science  
Wrocław University of Technology

**Abstract.** We present a voter verifiable Internet voting scheme which provides anonymity and eliminates the danger of vote selling even if the computer used by the voter cannot be fully trusted. The ballots cast remain anonymous - even the machine does not know the choice of the voter. It makes no sense to buy votes - the voter can cheat the buyer even if his machine cooperates with the buyer. Nevertheless, the voter can verify that his vote has been counted.

**Scope:** *Algorithms and data structures:* cryptography,

*Current challenges:* coercion-resistant internet voting

**Keywords:** electronic voting, vote selling, coercion resistance, anonymity

## 1 Introduction

Recently, there is a lot of public interest in electronic voting schemes. There are expectations that in a near future modern technologies may significantly improve the election procedures. However, while it became evident that traditional procedures have many inevitable flaws, it is still an unsolved problem how to design electronic voting schemes that fulfill all security demands. In this paper we concern the problem of casting a vote via Internet, which is the most challenging problem.

**Coercion-free voter-verifiable Voting Schemes** One can regard a (voter-verifiable) voting scheme as a process of submitting messages  $v(x_i)$  to a kind of bulletin board by voters  $x_1, \dots, x_N$  in such a way that

- every  $x_i$  can verify if  $v(x_i)$  is delivered to the bulletin board (voter verifiability),
- it is infeasible to link  $x_i$  with his vote; even if  $x_i$  is cooperating, it is infeasible to build a convincing proof that  $x_i$  voted in a particular way (coercion freeness).

**Motivations for Internet Voting** The first reason for introducing Internet voting is cost reduction. A growing fraction of the society has access to Internet, so one can try to use the existing infrastructure to reduce the costs and avoid manual work, which is the main cost factor in traditional schemes. For economical reasons, Internet voting is particularly interesting for countries with a low population density.

The second reason are the social costs of participation in elections. A person voting at a polling station is forced to get there, and this may cost time and money, and in some cases prohibit the voter to participate in the elections. The last factor becomes a growing problem in some countries. Internet voting may contribute to simplicity, flexibility and availability of voting.

The problems mentioned can be solved by mail-in voting, which becomes more and more popular in some countries. The dark side of mail-in voting are significant security

flaws that endanger the basic principles of democracy. Vote selling, blackmailing the voters, removing the ballots and adding new ones are significant problems that seem to be unsolvable for mail-in procedures.

**Voter Identification** In certain countries (like USA) the main practical problem is a reliable identification and authentication of voters. In other countries this is not a problem due to existing procedures of registration of inhabitants, advanced techniques implemented in ID cards and passports (e.g. in Malaysia). Biometric technology becomes mature and provides a high level of confidence for the election procedures. Moreover, price of biometric devices becomes affordable. Together with digital signatures this provides technical means that yield more reliable authentication than for the traditional voting procedures with manual checks.

**Problems and Risks of the Internet Voting** To some extent **anonymity** can be achieved by traditional voting on paper ballots. (Of course, there is no guarantee that the ballots do not contain hidden features that are invisible for the voter. In some political situations, even the threat that there might be such hidden features may prohibit to vote freely.) Electronic ballots are much harder to handle: if the ballots are identical, then there will be plenty of ways to attack the system by casting additional votes. If the ballots are unique, then they might be used for uncovering voters' preferences and for vote selling.

**Verifiability** of the election results is one of the major issues for electronic voting: while for the paper ballots there is a relatively reliable procedure preventing election frauds (as long as the commissions are honest), electronic voting is virtual and the voter may distrust the security mechanism of mixing and counting the votes. Therefore, one of the important features would be to provide the voter a (printed) trace that enables her to check that her vote has been counted and included in the final result. This approach of voting receipts is a central feature in many schemes (see for instance [2]).

**Vote selling** is the most important problem for Internet voting with profound consequences. Unlike in the case of the traditional voting process, buying votes might be very efficient, non-risky and does not require direct supervision of the buyer. Simply, the voter downloads and installs a special program that supervises his voting activities on his computer. This software sends appropriate information in an encrypted form to some remote server, even unknown to the voter selling a vote. Finally, the voter receives some reward - for instance in the form of digital cash, access codes to some Internet services or software products. One may try to secure the PC of the voter against such programs, but this seems to be a hopeless approach. An overwhelming majority of the users will not change the operating system or make affords to reconfigure it only for the sake of Internet voting. Another problem is that a single voter may want to sell a vote. In this case he will not implement the countermeasures or he will unmount them, if they are already deployed in the system. Necessary tools will be provided by the buyers of the votes.

**Scale of the Problems** Systems in which vote buying is easy are extremely dangerous. When we compare amount of money spent on election campaign and number of votes achieved, one can see that expenses per vote in some cases are higher than 40\$! So, from an economic point of view, it is reasonable to buy votes instead of launching an election campaign.

There are many documented cases of vote selling. For example, during the parliament election in Poland in 2005 one could buy votes in the Internet auction (a picture of a voting card taken with a digital camera was considered as a proof for casting a vote in the way expected). Similar cases were reported in Germany a few years earlier. One could buy mail-in votes in packages per 1000 and 10.000 ballots (!). There are cases reported of removing ballots in the case of mail-in voting. There is a famous example of the USA presidential election in 2004 in Duval County, Florida, where 58.000 of mail-in votes disappeared from a post office.

In many countries, there are cases reported that the number of invalid ballots is strongly correlated with the support for a particular candidate. This concerns the Bush-Kerry and Bush-Gore cases in USA [1, 23].

**Previous Solutions** Let us summarize the discussion above and point out problems with the previous solutions. Mail-in voting is a quite flexible and convenient system, so it becomes very popular in some countries (during the Bundestag elections in Germany in 2005, 25% of votes were mail-in votes). So, the influence of frauds could be significant in this case. The mail-in procedures are of questionable value for two reasons – it is perfect for vote selling and even worse, there is no way to verify correctness of the results (for instance the votes against a ruling party can be discarded).

Many electronic voting systems were proposed so far. Many of them are receipt-free [11, 18] and assume that the machines used for voting are honest. This approach seems to be unsuited for implementing electronic elections – one would require a detailed audit at least of the operating system and of the application used for voting. Such a verification of voter's hardware and software is practically impossible. Moreover, if a non negligible fraction of voters distrusts the system (even if it is honest), it should not be implemented for electronic elections.

Many Internet voting schemes allow a voter to cast a vote only once (or from a single machine). This makes vote selling very easy: a machine may have a special software installed that monitors voting activities and provides appropriate information to the buyer. A solution to this problem was implemented in the Estonian Internet voting system: a voter can revoke the previous ballot and cast a new one (this time in a traditional way). Each ballot is signed digitally by the voter, so it is possible to check which vote has to be removed (the signatures are removed before decryption of the ballots starts). The main problem of this system is that it provides no verifiability of the election results and that vote selling is possible.

A problem of untrusted voting machines can be solved with receipts. The first solution for which a voter gets a receipt proving that her vote was counted and at the same time it is meaningless for anybody else was presented by David Chaum [2]. In this case the voter becomes convinced about the election results, but at the same time she cannot sell her vote. Afterwards, other schemes with receipts were proposed. All these systems use a two stage verification. In the first stage, a voter can check that her vote appears a certain bulletin board. The second stage should convince her that her vote was properly processed by an array of mix-servers. Two major techniques are used for this purpose: Randomized Partial Checking [14] or Neff's zero knowledge proof procedures [21].

Recently, Klonowski *et al.* [17] proposed another scheme for voting machines. For this scheme each vote contains two parts, each part consists of two halves. One part

contains an encoded vote, the other part contains a random identifier. The halves of each part should appear after the final decoding, lack of any half is an evidence of a fraud during mixing and decoding. Each of the halves is processed separately and the processing servers cannot link them together until the final decoding. For this scheme a double verification is implemented:

- a voter can check that her vote identifier is included in the final bulletin board; so, she may be convinced that her vote is on the bulletin board as well,
- correctness of decoding and mixing is evidenced by the fact that there are two matching parts for each part of a vote.

In the systems [2, 17] the voting machine must be trusted to a certain degree - it knows the preferences of the voters; still, it cannot change them.

### 1.1 Properties of the New Scheme

We design an Internet voting system according to the following assumptions:

- the PC of a voter cannot be trusted,
- a voter may try to sell his vote, there are buyers ready to buy a vote,
- a voter should have an opportunity to convince herself that her vote was included in the final tally,
- a fraud attempt concerning a single vote should become detected with a constant probability, the malicious authority should be identified.

We design a protocol that generalizes the scheme from [17]. Let us list the main technical features of this protocol:

1. Each ballot is processed by a sequence of tallying authorities that perform mixing and partial decoding; if at least one of these tallying authorities is honest, then the vote remains anonymous.
2. While casting a vote the user obtains a receipt that can be used to check that his vote has been properly processed. If this is not the case for this single vote, then cheating can be detected with a fairly high probability and at least one of the cheating authorities can be identified.
3. The receipt and the transcript of the voting session on the computer of the voter do not suffice to determine the preferences of the voter. While casting a ballot the voter obtains a short message through an independent communication channel that is hidden for the machine used for voting.
4. A voter can change his decision by casting another ballot, which cancels the previous vote. Both ballots: the first one and the cancelling one appear in the final tally. Ballots are designed in such a way that they cannot be linked together.

It follows that we combine two properties that are somewhat contradictory: a voter can be convinced that his ballot has been counted, but simultaneously buying votes does not make sense. Indeed, even if the buyer supervises the computer of the voter (and can see what the voter is doing at the moment of casting a vote), he cannot be sure that the vote will not be revoked later. Moreover, in this case the voter can vote once more for another candidate or sell his vote to another party.

## 2 Mathematical Background

**RSA-RE Ciphertexts and Signatures** Now, we recall a construction of ciphertexts that may be signed and re-encrypted afterwards together with the signature. The idea is already used in the context of voting in [17], and comes from papers [8, 16]. The main advantage of re-encryption is that it allows instant verification of the mixing process without revealing any information about the contents of the ciphertexts and allowing checking the message origin.

**Key setup and ciphertext creation** Let  $N = pq$  be an RSA number, and let  $g$  be an arbitrary generator of a subgroup  $G \subseteq \mathbb{Z}_N^*$ , where  $G$  is a group with hard discrete logarithm problem. We skip the notation “mod  $N$ ” whenever operations within  $\mathbb{Z}_N$  are concerned.

The authority responsible for vote creation chooses  $e$ , which is co-prime with  $\varphi(N)$  and  $d$  such that  $e \cdot d = 1 \pmod{\varphi(N)}$ . Then  $d$  is the private signing key, whereas  $e$  is the public key for signature verification. An authority publishes  $\hat{g} = g^d$ .

Assume that each ballot has to be processed by  $\lambda$  mix servers before getting decrypted. For  $1 \leq j \leq \lambda$ , let  $y_j$  be the public key (for encryption) of the  $j$ th mix, and let  $x_j$  be the corresponding private key, where  $y_j = g^{x_j}$ . Every server obtains also a public key for signature verification, which is equal to  $\hat{y}_i = y_i^d$ .

In order to prepare a ciphertext we choose a string  $k_1$  uniformly at random. Then the ciphertext has the form:

$$(\alpha, \beta, \gamma, \delta) := (m \cdot (y_1 \cdot \dots \cdot y_\lambda)^{k_1}, g^{k_1}, m^d \cdot (\hat{y}_1 \cdot \dots \cdot \hat{y}_\lambda)^{k_1}, \hat{g}^{k_1}).$$

**Decoding process** When after some decoding and re-encryption such a ciphertext is delivered to mix  $i$ , it has the following form:

$$(\alpha_i, \beta_i, \gamma_i, \delta_i) = (m \cdot (y_i \cdot \dots \cdot y_\lambda)^{k_i}, g^{k_i}, m^d \cdot (\hat{y}_i \cdot \dots \cdot \hat{y}_\lambda)^{k_i}, \hat{g}^{k_i}).$$

We call it an onion since there are many “layers” of encryption and we have to remove these layers in order to decode it. Namely, the onion gets partially decrypted and re-encrypted – the following operations are executed with a randomly chosen  $r_i$ :

$$\begin{aligned} & (\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1}, \delta_{i+1}) := \\ & = (\alpha_i / \beta_i^{x_i} \cdot (y_{i+1} \cdot \dots \cdot y_\lambda)^{r_i}, \beta_i \cdot g^{r_i}, \gamma_i / \delta_i^{x_i} \cdot (\hat{y}_{i+1} \cdot \dots \cdot \hat{y}_\lambda)^{r_i}, \delta_i \cdot \hat{g}^{r_i}). \end{aligned}$$

It is easy to see that after performing these operations for  $k_{i+1} = k_i + r_i$  we get:

$$\begin{aligned} & (\alpha_{i+1}, \beta_{i+1}, \gamma_{i+1}, \delta_{i+1}) = \\ & = (m \cdot (y_{i+1} \cdot \dots \cdot y_\lambda)^{k_{i+1}}, g^{k_{i+1}}, m^d \cdot (\hat{y}_{i+1} \cdot \dots \cdot \hat{y}_\lambda)^{k_{i+1}}, \hat{g}^{k_{i+1}}). \end{aligned}$$

It should be clear that anybody can re-encrypt  $ue(m)$  in a similar way. For this purpose, only the knowing the public keys of servers is necessary.

**Signature verification:** If a RSA-RE-onion signature is correct, then for some  $k$  we have  $\alpha = m \cdot y^k$ ,  $\gamma = m^d \cdot \hat{y}^k$ , so  $\gamma = \alpha^d$ . Hence the verifier accepts the signature if and only if  $\alpha = \gamma^e$ .

**Notation:** One can see that first two parts of an onion,  $(\alpha, \beta)$  are ordinary ElGamal ciphertexts encrypted with the public key  $y_1 \cdot \dots \cdot y_\lambda$ . We will write  $ue(m)$  for a RSA-RE-onion of a message  $m$ , and  $e(m)$  for its first two components corresponding to an ElGamal ciphertext.

**Raising to a power** Let us observe that one can raise  $m$  hidden in  $ue(m)$  to an arbitrary power  $l$  without destroying the signature. Indeed:

$$ue(m)^l = (\alpha^l, \beta^l, \gamma^l, \delta^l) = (m^l \cdot (y_1 \cdot \dots \cdot y_\lambda)^{k \cdot l}, g^{k \cdot l}, m^{d \cdot l} \cdot (\hat{y}_1 \cdot \dots \cdot \hat{y}_\lambda)^{k \cdot l}, \hat{g}^{k \cdot l}).$$

The last expression is  $ue(m^l)$ , a RSA-RE-onion of a message  $m^l$ , with exponent  $k \cdot l$  used for encryption.

### 3 Building Blocks

Let us describe design of a voting card and a voting ballot which are used in our protocol. There are the following basic assumptions:

- There is a known list of possible voting options (list of candidates):  
 $o[0], o[1], \dots, o[K]$ .
- One of the options corresponds to an invalid vote, i.e.  $o[0] = \text{void}$ , to allow voters cast invalid votes.
- There is a fixed label  $p$  that will be used on the identifier card.

**Voting card** A *voting card* is the main building block of our scheme. Every card contains four parts (labeled  $A, B, C, D$ ) which have similar contents. For a card  $x$  a random permutation  $\pi$  over  $\{0, \dots, K\}$  is chosen independently at random. For the sake of the ease of use, we confine ourselves to random cyclic shifts, that is, we choose  $k$  at random and define  $\pi(j) = j + k \bmod (K + 1)$ . Using random cyclic shifts instead of random permutations seems to be:

- more handy – it is much easier to pass information about cyclic shift used in the card than the information about a permutation. Moreover, choosing a voting option in that case is simpler,
- more secure – a potential subliminal channel is much smaller.

For  $i = A, B, C, D$ , the part  $i$  of a card  $x$  contains the following values:

- a ciphertext  $e(r_i^x)$  of a random *header*  $r_i^x$ ,
- ciphertexts of identifiers  $o_i^x[0], \dots, o_i^x[K]$  listed in the order determined by  $\pi$ , namely  $e(o_i^x[\pi(0)]), \dots, e(o_i^x[\pi(K)])$ , the values of the identifiers are defined below,
- a list of ciphertexts  $e(v_i^x[0]), \dots, e(v_i^x[K])$  of random values  $v_i^x[0], \dots, v_i^x[K]$ .

All values contained in a card are encrypted with the version of ElGamal scheme discussed in the previous section. The public key used is the product of the public keys of all tallying authorities. A card can be depicted as it is presented on the Figure 1.

Additionally, a *proper* card  $x$  fulfills the following conditions:

Figure 1. A voting card  
voting card  $x$

$e(r_A^x)$	
$e(o_A^x[\pi(0)])$	$e(v_A^x[0])$
$e(o_A^x[\pi(1)])$	$e(v_A^x[1])$
$e(o_A^x[\pi(2)])$	$e(v_A^x[2])$
$e(o_A^x[\pi(3)])$	$e(v_A^x[3])$
$\dots$	$\dots$
$e(o_A^x[\pi(K)])$	$e(v_A^x[K])$
part A	

$e(r_B^x)$	
$e(o_B^x[\pi(0)])$	$e(v_B^x[0])$
$e(o_B^x[\pi(1)])$	$e(v_B^x[1])$
$e(o_B^x[\pi(2)])$	$e(v_B^x[2])$
$e(o_B^x[\pi(3)])$	$e(v_B^x[3])$
$\dots$	$\dots$
$e(o_B^x[\pi(K)])$	$e(v_B^x[K])$
part B	

$e(r_C^x)$	
$e(o_C^x[\pi(0)])$	$e(v_C^x[0])$
$e(o_C^x[\pi(1)])$	$e(v_C^x[1])$
$e(o_C^x[\pi(2)])$	$e(v_C^x[2])$
$e(o_C^x[\pi(3)])$	$e(v_C^x[3])$
$\dots$	$\dots$
$e(o_C^x[\pi(K)])$	$e(v_C^x[K])$
part C	

$e(r_D^x)$	
$e(o_D^x[\pi(0)])$	$e(v_D^x[0])$
$e(o_D^x[\pi(1)])$	$e(v_D^x[1])$
$e(o_D^x[\pi(2)])$	$e(v_D^x[2])$
$e(o_D^x[\pi(3)])$	$e(v_D^x[3])$
$\dots$	$\dots$
$e(o_D^x[\pi(K)])$	$e(v_D^x[K])$
part D	

- $r_A^x = r_B^x$  and  $r_C^x = r_D^x$ ,
- $\pi$  is a cyclic shift,
- for every  $k$  and  $i, j$ , we have  $o_i^x[k] = o_j^x[k]$  and  $v_i^x[k] = v_j^x[k]$

Recall that  $o[i]$  denotes an identifier of candidate  $i$  ( $o[0]$  serves as a void candidate). There are two types of proper cards used:

- If  $o_i[\pi(j)] = o[\pi(j)]$  for  $j \leq K$ , then we call it a *voting card*.
- If  $o_i[\pi(j)] = p$  for  $j \leq K$ , then we call it an *identifier card*.

A voter obtains  $n$  pairs of cards (where  $n$  is parameter chosen by a voter), each pair consists of a voting card and an identifier card posted in a random order.

**Voting ballot** A voting ballot is obtained from a voting card and an identifier card of the same pair. The following steps are performed in order to cast a vote (details are described in Section 4):

1. one of the rows on both cards is chosen,
2. the headers are modified so that the equal headers remain equal; similarly the values  $v_i^x$  get modified - the values from parts A and B remain equal, those from C and D become different,
3. all ciphertexts are re-encrypted,
4. the voting system attaches an RSA-RE-signature to each ciphertext,

More precisely, if  $ue(z)$  denotes  $e(z)$  after re-encryption and RSA-RE-signing by one of the Registration Servers and a voter has chosen row  $j$ , then a voting ballot has the form presented on the Figure 2a or, depending on the order of a voting and an identifier cards received from BGS, presented on the Figure 2b.

Finally, all parts A, B, C, D are signed by the voter, moreover, the parts C and D are encapsulated in a special ciphertext before delivering the ballot to the voting system (and will be used only in the case of a vote revocation).

**Infrastructure** The parts involved in the voting protocol are: tallying authorities, a Ballot Generation Server (BGS), responsible for generating cards, registration servers (RS) that are interfaces between the voters and the tallying authorities, a voter, say Alice, who uses an application APP on her PC.

Figure 2a. An example voting ballot

	part A	part B	part C	part D
u	$ue(r)$ $ue(o[\pi(j)])ue(v'_j)$	$ue(r)$ $ue(o[\pi(j)])ue(v'_j)$	$ue(s)$ $ue(o[\pi(j)])ue(\hat{v}_j)$	$ue(s)$ $ue(o[\pi(j)])ue(\bar{v}_j)$
l	$ue(s)$ $ue(p)ue(v'_j)$	$ue(s)$ $ue(p)ue(v'_j)$	$ue(t)$ $ue(p)ue(\hat{v}_j)$	$ue(t)$ $ue(p)ue(\bar{v}_j)$

Figure 2b. A different form of a voting ballot

	part A	part B	part C	part D
u	$ue(s)$ $ue(p)ue(v'_j)$	$ue(s)$ $ue(p)ue(v'_j)$	$ue(t)$ $ue(p)ue(\hat{v}_j)$	$ue(t)$ $ue(p)ue(\bar{v}_j)$
l	$ue(r)$ $ue(o[\pi(j)])ue(v'_j)$	$ue(r)$ $ue(o[\pi(j)])ue(v'_j)$	$ue(s)$ $ue(o[\pi(j)])ue(\hat{v}_j)$	$ue(s)$ $ue(o[\pi(j)])ue(\bar{v}_j)$

## 4 Voting Process

### Part I: Ballot Generation Procedure

This part of the protocol is executed in interaction between BGS, Alice and application called APP running on her PC.

1. Alice requests  $n \geq 10$  pairs of cards.
2. APP sends  $n$  requests to the BGS by an anonymous communication channel.
3. BGS responds for each request with the following data:
  - voting card, identifier card (in random order),
  - commitments to the identifiers contained in cards,
  - commitments to the values  $v_i^x[k]$  contained in cards,
  - commitments on the cyclic shifts used for constructing these cards,
  - digital signature under these cards.
4. Alice chooses a pair of cards for voting, let us say, a pair number  $i : 1 \leq i \leq n$ . All other cards will be used for verification.
5. APP sends an anonymous request to the BGS to reveal values of the identifiers, values  $v_x^y$  and cyclic shifts used in verification cards (every request should contain a digital signature of BGS under cards)
6. BGS checks if signatures are valid and responds (anonymously) with data requested
7. APP checks validity of the response – checks if received values satisfy commitments, if so then the following steps are executed:
8. Alice obtains information about the cyclic shift used in the cards chosen, obtains information about the identifier  $r_A$  of an identifier card (this information is sent through a channel that is inaccessible to the PC running APP (e.g. phone, SMS, ...))

The cards in a pair are transmitted in a random order, so neither the voter nor the voter's PC knows, which of the two cards is a voting card and which one is an identifier card. Also the cyclic shift used in the voting card remains hidden for the PC of Alice. Moreover, the probability that the shift declared by the BGS differs from the real one is equal to  $\frac{1}{n}$  and thus Alice can make it as small as desired (by getting more cards). Let us resume state of knowledge of the participants at the end of the Ballot Generation Procedure:

- BGS know which voting card (and identifier card) will be used by a voter. So in the following steps we will change form of the cyphertexts.
- Alice should be convinced (thanks to the steps 4, 5) that the shift used in her voting card corresponds to the obtained one.
- APP knows neither the contents of a voting card nor Alice's choice.

### Part II: Vote casting

The purpose of the following part of the protocol is not only to allow Alice to cast a vote, but also to change the appearance of parts of a voting card to keep an Alice's choice secret from BGS. Participants of this part of the procedure are Alice, the application APP on the PC of Alice, and a registration server RS.

1. Alice makes her choice - she chooses a row  $w$  of the cards according to the cyclic shift used and her voting preferences. Namely, if she chooses the  $j$ th candidate, then the row  $w$  has to contain a ciphertext of  $o[j]$  in the voting card, that is  $\pi(w) = j$ .

Let  $l$  and  $u$  be the cards chosen by the voter ( $l$  stands for the lower card,  $u$  stands for the upper card). From now on we skip the index  $w$  and use the notation  $r_{s,x}$  for  $r_s^x$ ,  $o_{s,x}$  for  $o_s^x[\pi(w)]$ , and  $v_{s,x}$  for  $v_s^x[\pi(w)]$ , for  $s = A, B, C, D$ ,  $x = u, l$ .

2. APP performs the following steps:
  - (2.1) it creates a ballot by selecting the values from the headers and the chosen row  $w$ . For  $Y = A, B, C, D$ , and  $x = l, u$  let  $Y_x = (e(r_{Y,x}), e(o_{Y,x}), e(v_{Y,x}))$ . Then two parts consisting of four blocks are formed:
    - $T_u = (A_u, B_u, C_u, D_u)$
    - $T_l = (A_l, B_l, C_l, D_l)$
  - (2.2) APP modifies the values  $r_{Y,x}$  and  $v_{Y,x}$  contained in  $T_u$  and  $T_l$ . For this purpose each plaintext of them is raised to a random power (the operation is performed on ciphertexts, as described before). For ciphertexts containing the same values (e.g.  $r_{A,x}$ ,  $r_{B,x}$ ) the same random powers are used - therefore these values remain equal. The only exception is for the ciphertexts encoding  $v_{Y,x}$  for  $Y = C, D$ ,  $x = u, l$  - for them the powers chosen should be different. The random powers used for modifying the headers  $r_{Y,x}$  are stored for the later use.
  - (2.3) APP sends  $T_u, T_l$  (after all modifications performed) to RS.
3. The RS performs the following steps:
  - (3.1) It modifies the values  $v_{Y,x}$  for  $Y = C, D$ ,  $x = u, l$ , by raising the ciphertexts to random powers. These powers must be different to ensure that the ciphertexts held so far in  $v_{C,x}$  and  $v_{D,x}$  become different in a way that is not known by APP.
  - (3.2) It signs all ciphertexts using RSA-RE-signature scheme,
  - (3.3) RS sends modified and signed ballots  $T_u, T_l$  back to APP.
4. APP performs the following steps:
  - (4.1) APP re-encrypts all ciphertexts (together with the signatures).
  - (4.2) APP performs a random permutation of the list  $A_u, B_u, A_l, B_l$ , the same steps are executed for  $C_u, D_u, C_l, D_l$ .
  - (4.3) APP contacts RS and obtains a challenge  $c$ .

- (4.4) The voter signs the challenge obtaining  $sig_v(c)$ , a deterministic signature scheme is used.
- (4.5) APP derives (in a deterministic way) an encryption key  $K := \mathbf{R}(sig_v(c))$  from a pseudorandom generator  $\mathbf{R}$ .
- (4.6) Together with the voter, APP prepares the following packets:
- (a)  $(A_u, A_l, B_u, B_l, P_{AB})_{sig(Alice)}$
  - (b)  $(Enc_K((C_u, C_l, D_u, D_l, P_{CD})_{sig(Alice)}))_{sig(Alice)}$
- where  $Z_{sig(Alice)}$  denotes  $Z$  together with the Alice's signature attached to  $Z$ , and  $Enc_K$  denotes symmetric encryption with the key  $K$ . We say that the second packet contains a *revocation code*. APP sends both packets to RS.
- (4.7) RS checks the signatures and the correctness of the first packet according to  $P_{AB}$ . The first packet is stored in the set of votes cast, the second (encrypted) packet is stored in a repository of revocation codes. RS also provides a receipt for the voter which is a signature of RS under both packets.

**Properties of ballots** Before we discuss how the votes are revoked and counted let us discuss basic properties of the procedure of creating the ballots:

- A ballot prepared by Alice and APP together with RS consists of two sets of ciphertexts - each originating from a different card. Neither Alice nor RS can separate the ciphertexts corresponding to the voting card. This prevents removal of Alice's vote. (The remaining ciphertexts can be used to check correctness of vote counting and detect a malicious authority if some ciphertexts are missing).
- Each half of the ballot contains two parts: one composed from parts  $A$  and  $B$  used for voting or detecting manipulations, the parts  $C$  and  $D$  are used for composing a revocation code (again, one part is responsible for revocation itself, while the other part guards against manipulations).
- The revocation code is encapsulated in a ciphertext that cannot be opened by RS until Alice provides the key by signing the challenge. Alice need not to use the PC on which application APP was running - the challenge can be presented to Alice by RS on her demand.
- The revocation code cannot be distinguished from the codes constructed from parts  $A$  and  $B$ , until it is fully decrypted.
- The values  $v_{x,i}$  contained in the votes are not known to anybody until the final decryption.
- The parts  $A$  and  $B$  ( $C$  and  $D$ , respectively) of a ballot composed from the same card contain the same header. So, if during counting procedure no part is removed, then in the final result each header occurs exactly twice.
- The header of parts  $A$  and  $B$  of an identifier card is known to the voter. Originally it is known to BGS. Then Alice get informed about the header and she raises it to a random power. The remaining headers (including the header of parts  $C$  and  $D$  of the voting card used for revocation) are not known to anybody - the initial values set by BGS are modified at random.

**Vote revocation** A voter can cancel his previous vote by signing a challenge which was used during the previous vote casting. The procedure of voting for the second (third, ...) time contains an additional step. It is a fair exchange between a commission and a voter.

The voter sends a signature for the challenge used during the previous vote generation to the commission. The commission uses it for decryption of the revocation vote, posts it on the Bulletin Board number 0 and increments the number of the canceled votes.

**Tallying process** After closing the polling stations the RS servers are closed as well. For the process considered below each of the parts  $A, B, C, D$  is called a block. Note that each block consists of three signed ciphertexts. During the procedure described below, each block is processed together.

First, each RS mixes all its blocks and sends them to the Bulletin Board with the index 0. Now, the mixing procedure is executed by an array of mix servers run by independent tallying authorities. For  $1 \leq i \leq \lambda$ , the  $i$ th tallying authority runs a server that executes the following steps:

- it reads the blocks from Bulletin Board  $i - 1$  and checks the signatures,
- it partially decodes the ciphertexts in each block with its private key,
- it re-encrypts each ciphertext,
- it mixes the blocks at random and sends them to the Bulletin Board  $i$ .

The last tallying authority gets, after decryption, plaintexts of the ciphertexts included in the blocks. It presents them in the Bulletin Board  $\lambda$  together with a Zero Knowledge Proof of correct decoding. Now, on the  $\lambda$ th-Bulletin Board one can find election results. Namely, it contains the triples of the form:  $(r, o[i], v)$ , and  $(s, p, v')$ . For the sake of convenience, we sort them in a lexicographic way.

**Vote Counting and Checking Correctness** First we check that for each  $r$  there are two triples of the form  $(r, -, -)$  or no such a triple. If this is not the case that somebody is cheating and an investigation starts. The algorithm used can be borrowed from the scheme from [17].

Now assume that the following triples appear on the bulletin board:  $(r, e, v), (r, e', v')$ . If  $e \neq e'$ , then something went wrong (and the same random header occurred for two votes). So assume that  $e = e'$ . Now, the following cases are possible:

- $e = o[i]$ , that is,  $e$  is one of the voting options then:
  - if  $v = v'$ , then it is vote for  $o[i]$ ,
  - if  $v \neq v'$ , then it is a revocation of a vote for  $o[i]$ .
- $e = p$ , then:
  - if  $v = v'$ , then  $r$  is a vote identifier (which can be controlled by some voter)
  - if  $v \neq v'$ , then  $r$  is an anti-vote identifier (it can be controlled by a voter together with RS, if RS reveals the exponent used to modify the header.

## 5 Conclusions

We have presented an Internet voting scheme which is coercion-free without assumption of certified software on a secure machine used in a voting process. The diagram in the Appendix contains a short comparison between the existing voting schemes.

## References

1. Agresti, A., Presnell, B.: Misvotes, Undervotes and Overvotes: The 2000 Presidential Election in Florida. *Statist. Sci.* 17,4 (2002) 436-440.
2. Chaum, D.: Secret-Ballot Receipts and Transparent Integrity. Better and less-costly Electronic Voting and Polling Places. *IEEE S&P'04*.
3. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM* 24(2), 84-88, 1981.
4. Chaum, D., Ryan, P. Y. A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. *ESORICS '2005*, LNCS 3679, 118-139.
5. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Advances in Cryptology – CRYPTO '86*, LNCS 263, 186-194.
6. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. *Advances in Cryptology- CRYPTO '2001*, LNCS 2139, 368-387.
7. Gomułkiewicz, M., Klonowski, M., Kutylowski, M.: Rapid Mixing and Security of Chaum's Visual Electronic Voting. *ESORICS'2004*, LNCS 2808, 132-145.
8. Golle, P.: Reputable Mix Networks. *Privacy Enhancing Technologies (PET) 2004*, LNCS 3424, 51-62.
9. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-encryption for Mixnets. *CT-RSA '2004*, 163-178.
10. Hirth, M.: Receipt-Free K-out-of-L Voting based on ElGamal Encryption. *Workshop on Frontiers in Electronic Elections 2005*.
11. Hirth, M., Sako, K.: Receipt-Free Electronic Auction Schemes Using Homomorphic Encryption. *Information Security and Cryptology - ICISC 2003*.
12. Jakobsson, M.: A Practical Mix. *Advances in Cryptology- EUROCRYPT '1998*, LNCS 1403, 448-461.
13. Jakobsson, M.: Flash Mixing. *ACM Symposium on Principles of Distributed Computing '1999*, 83-89.
14. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. *USENIX Security Symposium '2002*, 339-353.
15. Karlof, C., Sastry, N., Wagner, D.: Cryptographic Voting Protocols: a Systems Perspective. *USENIX Security Symposium '2005*, 33-50.
16. Klonowski, M., Kutylowski, M., Lauks, A., Zagórski, F.: Universal Re-encryption of Signatures and Controlling Anonymous Information Flow. *Wartacrypt 2004*.
17. Klonowski, M., Kutylowski, M., Lauks, A., Zagórski, F.: A Practical Voting Scheme with Receipts. *International Security Conference (ISC)'2005*, LNCS 3650, 380-393.
18. Lee, B., Kim, K.: Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. *Information Security and Cryptology - ICISC 2002*, LNCS 2587, 389-406.
19. Mitomo, M., Kurosawa, K.: Attack for Flash MIX. *Advances in Cryptology- ASIACRYPT '2000*, LNCS 1976, 192-204.
20. McGaley, M.: Report on DIMACS Workshop on Electronic Voting - Theory and Practice, [http://dimacs.rutgers.edu/SpecialYears/2003/\\_CSIP/reports.html](http://dimacs.rutgers.edu/SpecialYears/2003/_CSIP/reports.html)
21. Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting. *ACM Conference on Computer and Communications Security '2001*, 116-125.
22. Rivest, L.R.: voting resources page, <http://theory.lcs.mit.edu/~rivest/voting/>
23. Smith, W. D.: Cryptography Meets Voting. <http://www.math.temple.edu/~wds/homepage/cryptovot.pdf>

## 6 Appendix

Elections →	traditional	electronic [17]	mail-in	Internet (Estonia)	our scheme
trusted parties	members of each polling station committee	voting machine, at least one tallying authority	whole post system, central counting commission	application, operator, central counting commission	at least one tallying authority
verification process	summing up partial results	verifiable receipts	none	none	verifiable receipts
who can cast additional votes	local commissions	local commissions	central commission	central commission	nobody
who can remove/invalidate votes	local commissions	nobody	postman, ...	central commission	nobody
level of anonymity (anonymity set)	local commission	local commission	full	?	full
vote selling	possible	impossible	very easy	impossible	impossible