Verifiable Internet Voting Solving Secure Platform Problem

Mirosław Kutyłowski, 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 verity that his vote has been counted.
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 current 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 – not only on the level of a cryptographic protocol, but also concerning the voting equipment as a potential adversary. In this paper we concern the problem of casting a vote via Internet, which is the most challenging problem.

Motivations for Internet Voting A growing fraction of voters has access to Internet, so one can try to use this infrastructure to avoid costly manual work of traditional schemes. The second reason for Internet voting 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. This becomes a growing problem in some countries and yields concerns about fairness of democratic elections, since for some social groups the actual cost of election participation (time loss, inconvenience, travel costs) is prohibitively high. The practical consequence of this situation is that economically most active voters do not participate in elections.

The problems concerned above can be solved by postal voting, which becomes increasingly popular. A dark side of postal voting are significant security flaws that endanger the basic principles of democracy. Vote selling, blackmailing the voters, removing the ballots and adding new ones seem to be unsolvable problems for postal voting.

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, and advanced features of ID cards and passports (e.g. in Malaysia). Together with digital signatures this provides affordable technical means that yield more reliable authentication than 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 longer guarantee that the ballots do not contain hidden features invisible for the voter.) 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. If they contain random values, then these values may be used to leak secrets. Another practical issue for Internet voting is voter authentication, but we assume that the voters can authenticate themselves with digital signatures.

Verifiability of the election results is one of the major issues for electronic voting: while for the paper ballots there are some procedures against election frauds (they work 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 concept of voting receipts is a central feature in many schemes (see for instance [2]). However, it is also a major problem for system design. At the same time two requirements should be satisfied:

– a receipt must convince a voter that her ballot was properly counted,
– a receipt must not reveal voter's choice.

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 performed with no 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 unknown remote server. Finally, the voter receives some reward. One may try to guard voter's PC against such programs, but this seems to be hopeless. Even if it would be possible for today's operating systems (it is not), an overwhelming majority of the users will not change the operating system or make efforts to reconfigure it only for the sake of Internet voting. Moreover, a voter may want to sell a vote. In this case he

would not install security system or he would unmount them, if they were already deployed. Necessary tools would be provided by vote buyers.

Systems in which vote buying is easy are extremely dangerous due to the cost of potential vote-buying systems compared to the cost of election campaign.

(Un)trusted Platforms Voting software and hardware are practically *black boxes*. Even if they have been checked, the voters has no real guarantee that the systems checked are identical with the systems installed on the voters' PC. Even if it is so, the voter may distrust the authorities involved in the audits. In particular, one may fear that the authorities could have installed some features for supervising a voter or manipulating his or her vote.

Previous Solutions Many electronic voting systems have been proposed so far. One can classify them into three groups:

- **Paper Based** - a voter votes in a booth on a paper ballot which is scanned and then electronically counted by the system [3, 5, 24],
- **Machine Based** - in a booth a voter operates some voting machine [2, 18, 22, 23],
- **Internet Based** - a voter uses her PC to cast a vote [12, 19].

Many schemes are receipt-free [12, 19], but assume that that the machines used for voting are honest. This approach seems to be unsuited for Internet voting – 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 infeasible.

In [11] a model of electronic voting schemes have been introduced. In the same paper, a coercion-resistant voting system was presented. Still, presented scheme does not take into account that cryptographic operations cannot be performed by a a voter herself ("Voter V_i includes non-interactive proofs of knowledge of $\sigma_i \cdots$"), so in fact, indirectly assumes honesty of the hardware and software used by the voter.

The problem of untrusted voting machines can be partially solved with receipts. Chaum [2] presented a 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. Hence the voter cannot convince a buyer about the vote cast. Later other schemes with receipts were proposed. All they use a two stage verification. First, a voter can check that her vote appears on 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 here: Randomized Partial Checking [15] and Neff's zero knowledge proof procedures [23].

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 reports them to the buyer. A solution of this problem was implemented in Estonia: a voter can revoke her electronic ballot and cast a traditional paper ballot. Each electronic ballot is signed digitally by the voter, so it is possible to check which ballot 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 and that vote buying still makes sense (only a fraction of sellers will revoke the vote).

Klonowski *et al.* [18] proposed a quite different scheme for voting machines. 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. This scheme implements double verification:
(a) 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,
(b) correctness of decoding and mixing is evidenced by the fact that there are two matching parts for each part of a vote.
In case of [2, 18] the voting machines must be trusted to a certain degree - they know the preferences of the voters; still, they cannot change them. No solution that would work regardless of dishonest voter's PC has been designed so far. Current research concentrates rather on providing a secure environment (like Trusted Platform proposals) - however this approach has basic limitations.

Internet Voting - Threats: Let APP denote the voting application run on the PC of a voter. For Internet voting protocols, there are three main types of threats:
Type A: A voter can be cheated by APP:
(1) APP can cast a vote for a different *specified* candidate,
(2) APP can cast a vote for a random candidate,
(3) APP can cast an invalid vote.
Type B: A voter may want to sell a vote:
(1) a vote buyer/coercer is physically present when a voter is casting a vote via Internet or he can impersonate the voter,
(2) the buyer/coercer provides appropriate application to be run on a machine used for vote casting (to monitor a voter's choice).
Type C: A voter can be cheated by a voting system that implements gathering the ballots, decoding them and computing the results. This problem concerns the protocols which do not provide verifiability:
(1) lack of global verifiability (schemes without audit procedures),

(2) lack of local verifiability (schemes which does not allow a voter to check if her vote was counted as intended).

From a practical point of view the threats of type A and type C are much different. In case of a system concerned in point C detailed audit procedures may take place. On the other hand, there is no control over voter's PC used for Internet voting. In particular, the voter can use any software as APP that yields an output according to the specification of the protocol.

Our Results We design a protocol that generalizes the scheme from [18]. Let us list the main technical features of our proposal:

1. Each encoded ballot is processed through 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 digital receipt that can be used to check if his vote has been properly processed. If a single ballot has been manipulated, then it become 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. Before 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 canceling one appear in the final tally, but 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. Even if the buyer supervises the computer of the voter (and can see what the voter is doing on this PC), he cannot be sure that the vote will not be revoked later from another machine.

2 Background of Encoding Techniques

RSA-RE Ciphertexts and Signatures Let us recall a construction of ciphertexts that may be signed and re-encrypted later together with the signature. The idea has been already used for e-voting [18], and originates from [9, 17]. The main advantage of re-encryption is that it allows certain instant verification of the mixing process without revealing anything about the plaintexts.

Key Setup and Ciphertext Creation Let $N = pq$ be an RSA number, and let g be an arbitrary generator of a cyclic subgroup $G \subseteq \mathbb{Z}_N^*$ such that discrete logarithm problem is hard in G. We skip the notation "$\mathrm{mod}\, N$" 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 $e \cdot d = 1 \bmod \text{lcm}(p-1, q-1)$. Then d is the private signing key, whereas e is the public key for signature verification. The key $\hat{g} = g^d$ is published.

Assume that each ballot has to be processed by λ mix servers before getting decrypted. For $1 \leq j \leq \lambda$, let y_j be the public key (for encryption) of the jth 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 $\hat{y}_i = y_i^d$.

A ciphertext of m is created with a randomly chosen k_1 and has the following form:

$$(\alpha, \beta, \gamma, \delta) := (m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^{k_1}, g^{k_1}, m^d \cdot (\hat{y}_1 \cdot \ldots \cdot \hat{y}_\lambda)^{k_1}, \hat{g}^{k_1}) \; . \quad (1)$$

We call it an RSA-RE-onion, since there are many "layers" of encryption and we have to remove these layers in order to decode it.

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

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

The following operations are executed with r_i chosen at random:

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

It is easy to see that we get the following tuple with $k_{i+1} = k_i + r_i$:

$$(m \cdot (y_{i+1} \cdot \ldots \cdot y_\lambda)^{k_{i+1}}, g^{k_{i+1}}, m^d \cdot (\hat{y}_{i+1} \cdot \ldots \cdot \hat{y}_\lambda)^{k_{i+1}}, \hat{g}^{k_{i+1}}) \; .$$

Observe that for re-encryption we need to know only the public keys concerned.

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 $\gamma = \alpha^d$. Similarly, $\delta = \beta^d$. So we say that the outcome of verification is positive iff $\alpha = \gamma^e$ and $\beta = \delta^e$.

Notation Obviously, the first two parts of an onion, (α, β), is a regular ElGamal ciphertext obtained for the public key $y_1 \cdot \ldots \cdot y_\lambda$. 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:

$$ue(m)^l = (m^l \cdot (y_1 \cdot \ldots \cdot y_\lambda)^{k \cdot l}, g^{k \cdot l}, m^{d \cdot l} \cdot (\hat{y}_1 \cdot \ldots \cdot \hat{y}_\lambda)^{k \cdot l}, \hat{g}^{k \cdot l}) \; .$$

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

Opening an Onion It can be checked that an onion $(\alpha, \beta, \gamma, \delta)$ has been created according to formula (1) by *opening* it. Namely, the party that has created the onion reveals the exponent k_1 used. Note that no private keys are necessary.

Coupled Ciphertexts Assume that two ciphertexts for two different recipients have to be processed together, but in the meantime they have to be re-encrypted. If we do it in a standard way, then the link between the ciphertexts considered is lost after re-encryption and one could replace one of these ciphertexts. However, we can *couple* the ciphertexts of m_1 and m_2 by constructing:

$$(m_1 \cdot \alpha^k, m_2 \cdot \beta^k, g^k) \ ,$$

where k is chosen at random, and α, β are the public keys of the addressees of the ciphertexts. We can re-encrypt such a ciphertext by multiplying its components by, respectively, α^h, β^h, g^h for a random h. It is easy to see that use of the same k for encrypting m_1 and m_2 does not introduce any security risks, since a procedure to break the scheme (even with a private key corresponding to β) would yield a method to break ElGamal ciphertexts created with α.

3 Scheme Construction

Since the number of details in the final scheme might be confusing, we present a family of schemes, each time discussing improvements introduced. This should explain the final construction; in the next section, one can find a (high level) step-by-step description of the protocol.

Model The physical setup of the system is following. Voters use their PC's which are connected to Internet. PC's are running a voting application - APP. PC's are also equipped with smart card readers for implementing trustworthy digital signatures. A specialized server named BGS (Ballot Generation Server) is responsible for preparing ballots. A registration server named RS is responsible for verifying signatures of the voters. There are public bulletin boards of authorities responsible for mixing and decoding votes.

Version 0 - Estonian like Solution Let $y_1, \ldots, y_\lambda$ denote the public keys of the authorities responsible for mixing and decoding the ballots. APP prepares a ballot containing m as an ElGamal ciphertext

$$(m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^k, g^k)$$

for a random k. Such an encrypted ballot, signed digitally by the voter, is sent to the bulletin board. At the end of an election day: each signature is verified and it is checked that no voter cast more than one vote. Then decoding and mixing the ballots is performed by the tallying authorities after stripping off the signatures. Correctness of decoding should be checked, e.g. with RPC ([15]). An electronic ballot can be revoked by the voter in a polling station before the end of the election day.

Advantages: *(A.0.1)* If at least one tallying authority is honest, then voter's preferences remain hidden.

(A.0.2) A buyer cannot be sure that the voter will not revoke a vote after selling it (however, the buyer may pay after checking that the buyer has not revoked his e-ballot).

Disadvantages: *(D.0.1)* All threats of type A apply. A fraud cannot be detected, since the random exponent k should be erased immediately in order to protect voter's privacy.

(D.0.2) Practically, the threats of type B apply - revoking personally is tedious and indicates that the voter might be a (dishonest) vote seller.

(D.0.3) Local verification is impossible (threat C.2), only global verifiability can be implemented.

Version 1 - Locally Verifiable Scheme

We apply an idea from [18]. A ballot consists of four ciphertexts: two of them encode the voter's choice (just the identifier of the candidate, the same for all ballots), the next two encode an identifier known to a voter and used for verification. The order of the ciphertexts in the ballot is random. After the first decoding and re-encryption all ciphertexts are permuted at random so that the link between the ciphertexts from the same ballot is lost.

Advantages: All advantages of Version 0 are preserved. Additionally, the scheme is immune against threats of type C:

(A.1.1) Ballot decoding process is in some extent locally verifiable - a voter can check that the identifier of his ballot appears on the final list.

(A.1.2) Removing, modifying and adding one vote can be detected with probability at least $\frac{5}{6}$ in the case of the first tallying authority – equal to the probability of removing 2 parts encoding a vote without removing 2 parts encoding an identifier ($= 1 - \frac{2}{4} \cdot \frac{1}{3}$). The probability that m manipulated ballots remain undetected is lower than $(\frac{1}{4W})^m$ in case of the remaining authorities, where W is the number of votes cast [18].

Disadvantages: The scheme is not immune against threats of type A and B.

Version 2 - Securing the Scheme against a Dishonest APP

Let us assume that there is N candidates. The ballots are created by an independent authority, so called *Ballot Generating Server (BGS)*; the following steps are executed in order to create a single ballot:

(1) BGS generates N 4-tuples of onions, where each tuple is generated as for Version 1 of the protocol and the ith tuple encodes a vote for the ith candidate.

(2) The list of 4-tuples is shifted circularly by a random shift s.

(3) BGS sends the tuples to the PC of a voter.

(4) BGS sends the shift s through an independent channel directly to the voter.

(5) The voter indicates which of the 4-tuples should be cast as the vote.

(6) APP re-encrypts the tuple chosen, permutes at random the onions obtained

and presents them to the voter for signing.
(7) The voter signs the onions and APP sends them to the bulletin board.

Advantages: The scheme is still immune against threats of type C. Additionally:
(A.2.1) Voter's PC does not know voter's preferences (as long as PC and BGS do not collude) – it is the first step to eliminate threats of type A.1.

Disadvantages: *(D.2.1)* PC can change the vote at random - it can send for signing a different tuple than indicated by the voter. Nothing will be detected. However, APP cannot change the vote to a vote for a chosen candidate.
(D.2.2) The voter has to trust BGS - the threats of type C apply in case of BGS.
(D.2.3) APP may pretend that it is taking onions from BGS, but in fact generate them itself. So all threats of type A still apply.

Version 3 - Enforcing Cooperation with BGS
Instead of onions, BGS creates RSA-RE-onions. Two ciphertexts from a 4-tuple indicating a vote for a candidate i encode the plaintexts c_i and c_i', where c_i, c_i' are chosen at random from G for all ballots (so in particular, the discrete logarithms of c_i and c_i' with respect to g are unknown).

Advantages: *(A.3.1)* At any level of decoding it can be checked that an onion originates from BGS (by verifying signature that will be inserted into an onion). Hence APP cannot create own onions. Together with the property (A.2.1) it prevents threats of type A.1.
(A.3.2) The RSA-RE onions do not eliminate manipulations through raising to power. However, it is unknown how to make a valid vote for another candidate from a vote for candidate i. Together with property (A.2.1) it guarantees that APP cannot create random valid votes. So threats of type A.2 are prevented.

Disadvantages: The scheme is immune against threats of type B.2, but threats of type B.1 still apply. Threats of type C exist with respect to BGS.

Version 4 - Securing the Scheme against Malicious BGS
In order to enforce honest behavior of the BGS, the following additional steps are executed:
(1) Instead of one *card* containing N 4-tuples, two cards are presented on request by the BGS to APP (optionally more than two). The onions are still not composed as RSA-RE-onions (there is no signature inside). However, each card is signed digitally in a conventional way for non-repudiation purposes.
(2) Together with the cards APP receives cryptographic commitments of:
    (a) the shifts used for both cards,
    (b) the identifiers.
The commitments can be printed by the voter or mailed in a traditional way on

scratch cards.
(3) The shifts are sent to the voter through a secure channel (without involvement of her PC).
(4) The voter chooses one of the cards for checking.
(5) BGS responds with values opening the commitments and onions for the card requested. Moreover, BGS creates RSA-RE-onions from the onions of the second card.
(6) APP presents the result of opening to the voter, who checks the results.

Advantages: As Version 3, it is immune against threats of types A.1, A.2, B.2. Additionally:
(A.4.1) Even a single fraud of BGS becomes detected with a constant probability. So the scheme is immune against threats of type C as long as BGS and APP do not cooperate.

Disadvantages: Still, the scheme does not prevent threats of type B.1 and A.3. Unfortunately, this solution enables effective vote selling (B.2): A vote's seller can input obtained shifts to APP, then APP (not the voter!) chooses a card for voting and verifies the shifts obtained from the voter once BGS reveals the shifts from the unused cards.

Version 4a - Securing against vote seller BGS, generates s_1, s_2, and derives shift, as $s = s_1 + s_2$. Instead of sending commitment of s to an APP, BGS sends commitments of s_1 and s_2. At the same time, BGS sends values of s, s_1, s_2 of each card to a voter. For each card only one of the commitments of values s_1, s_2 is revealed to APP. The choice is made deterministically, based on a deterministic signature of a voter under a publicly known value z.

Advantages: Now, a voter cannot prove to APP her shifts, because she can check in advance which shift-components will be revealed by BGS. Still, BGS cannot cheat, because it does not know the voter's signature under z. It eliminates threat $B.2$.

Version 5 - Ballot Revoking
Instead of a set of N 4-tuples (each tuple for a different candidate) we have two 4-tuples for each candidate: a *voting tuple* and a *revoking tuple*. The voting tuple for candidate i contains ciphertexts of c_i and c'_i, while two ciphertexts of the revoking tuple for candidate i encode d_i and d'_i, where d_i, d'_i are chosen similarly as c_i and c'_i and serve as "anti-votes" canceling c_i, c'_i. The identifiers in the revoking tuple are chosen independently of the identifiers contained in the voting tuple.

For the purpose of revoking the following additional steps are executed:
(1) *Registration Server* (RS) sends a random challenge r to APP,

(2) APP derives a symmetric key K from the voter's signature of r (a deterministic scheme like RSA has to be used),
(3) APP re-encrypts the revoking tuple and encrypts it with K, the resulting ciphertext is signed by the voter,
(4) APP sends the ciphertext of the revoking tuple together with the voter's signature to RS.

At a due time, when the voter decides to revoke his ballot, the following steps are executed (the voter may use a different PC):
(1) the voter signs r once more and sends the signature to RS,
(2) the RS checks the signature, derives K, retrieves the revoking tuple and puts it on the bulletin board just like a voting tuple.
Advantages: *(A.5.1)* Even if a buyer is present when a vote is cast, he cannot be sure if a voter does not revoke this vote and casts another vote later (from a different PC). So the threat type B.1 is minimized.

Disadvantages: Still, the scheme does not prevent threats of type A.3. Moreover, APP may cheat and send to the bulletin board a different revoking tuple than it should. In particular, a voter may create own APP in order to revoke votes for a candidate, she hates.

Version 6 - Enforcing APP to play fair
Version 5 of the algorithm requires a pair of 4-tuples per candidate in a ballot obtained from BGS. We change it: now we have two pairs of 4-tuples with identical contents for each candidate - let us call them the *lower* and the *upper* pair. To each pair, BGS attaches two encrypted *marks*, the first mark is either A or $\overline{A}$, the second *hidden* mark is either X or $\overline{X}$. For each candidate the BGS assigns mark A to the upper pair and mark $\overline{A}$ to the lower pair or vice verse, each option with probability $\frac{1}{2}$, independently of the choices for other candidates. Similarly, BGS assigns hidden marks X and $\overline{X}$ to the pairs, independently from other choices.

Now let us describe how to attach the marks chosen for each pair. We use two public keys of the bulletin board, say γ and χ. Then to each onion from a pair of 4-tuples we attach the marks chosen. Namely, instead of an ElGamal ciphertext $(m \cdot (y_1 \cdot \cdot y_\lambda)^k, g^k)$, BGS creates

$$(m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^k, M_1 \cdot \gamma^k, M_2 \cdot \chi^k, g^k)$$

where M_1 and M_2 are the marks chosen for this pair. (In fact, the RSA-RE signatures have to be attached to the resulting tuple.) As already seen, such a coupled ciphertext can be re-encrypted.

The following additional steps are executed while casting a vote:
(1) Together with commitments the voter obtains information about configuration of A and $\overline{A}$ marks in the ballot (and no information about configuration of

X and $\overline{X}$ marks).
(2) After choosing a candidate the voter indicates which pair of marked 4-tuples to use: the upper or the lower pair corresponding to her candidate. The ciphertexts chosen are re-encrypted, signed traditionally and sent to RS. The marked revoking tuple is encrypted with the key K, and sent to RS as before.
(3) RS decrypts the first mark (either A or $\overline{A}$) and attaches it to the voting tuple published on the first bulletin board.
(4) The voter checks whether the right mark is published together with her voting tuple.
When a voter demands to revoke her vote, the following additional steps are executed:
(1) the bulletin board decrypts the second mark from the voting tuple obtained from the voter,
(2) after decrypting the ciphertext containing the revoking tuple the bulletin board retrieves both marks attached. The first mark is published together with the revoking tuple.
(3) The bulletin board checks whether both marks are the same for the voting tuple and for the revoking tuple. If not, then a fraud of APP is detected.

Advantages: *(A.6.1)* Since there is a voting tuple with mark A (respectively, $\overline{A}$) for each candidate, marks published on the bulletin board do not betray the choices of the voters even to BGS.
(A.6.2) If APP performs the steps of the protocol for a different candidate than indicated by the voter (as described by threat A.2), then it has to choose either the upper or the lower pair of this candidate. With probability $\frac{1}{2}$ the chosen pair is wrong- it has a mark different than expected by the voter. So APP cannot send a random vote instead of the one chosen by the voter without a fair chance of being detected.
(A.6.3) The voter may collude with APP and indicate a revoking tuple for a different candidate than chosen for a voting tuple. The voter can choose a tuple with the correct first mark. However, the voter does not know the hidden mark, so with probability $\frac{1}{2}$ the fraud attempt will be discovered while revoking the vote. Of course, we can increase the chances to detect a fraud: instead of a lower and an upper pair, we might have 2^k pairs for each candidate, where each k-bit mark is assigned to exactly one pair. This decreases the chances of an unnoticed fraud to 2^{-k}.

4 Scheme Construction - Details

Voter's point of view Although, a system may seem to be complicated (in fact it is true, if one wishes to fully understand security mechanisms), it is quite easy to be used by an average voter, a voter Alice performs the following steps:

- downloads and installs APP,
- chooses the number of cards to be downloaded from BGS,
- obtains by the independent channel values of card's ids, and component of shifts $s1, s2$ for each card,
- chooses a card for voting,
- chooses a row and a candidate,
- inserts her ID card with signing function into a reader and signs a ballot,
- obtains an electronic copy of her "vote".

When elections are closed, Alice can check, if her identifier is on the final bulletin board. If Alice revokes a vote cast or checks BGS, then a few additional steps are performed.

4.1 Building Blocks

During system setup we choose $c_0, c_1, \ldots, c_{N-1}, c'_0, c'_1, \ldots, c'_{N-1}$ – the values corresponding to the voting options (e.g. list of candidates). Similarly, c_N, $c_{N+1}, \ldots, c_{2N}, c'_N, c'_{N+1}, \ldots, c'_{2N}$ is the list of the values corresponding to the anti-votes, namely anti-vote c_{N+m} or c'_{N+m} corresponds to the vote for candidate/option m. One of the values corresponds to an invalid vote, i.e. $c_0 = \textit{void} = c'_0$, to allow voters cast invalid votes.

Card For every card, random s_1, s_2 define a random cyclic shift: $\pi(j) = j + s \bmod N$, for $s = s_1 + s_2$. Two pairs of random identifiers are used: (id, id') and (rid, rid'). The identifiers are coupled in some way (for instance $id' = \text{sign}_{BGS}(id)$, $rid' = \text{sign}_{BGS}(rid)$).

The jth row of a card contains an upper and a lower part. Each of them contains six ciphertexts for casting a vote and six ciphertexts for revoking a vote. Each 6-tuple contains 2 ciphertexts of marks: either of A or of $\overline{A}$ and either of X or of $\overline{X}$. The choice of marks is indicated by random N-bit sequences T_A and T_X. For $t \in \{A, X\}$,

$$f_{u,t,j} = \begin{cases} t & \text{if } T_t(j) = 1 \\ \bar{t} & \text{if } T_t(j) = 0 \end{cases}$$

For the lower part, $f_{l,t,j} \in \{t, \bar{t}\}$ and is different from $f_{u,t,j}$.
For $x \in \{u, l\}$ (u, l stand for *upper* and *lower*), row i contains the following 6-tuples for voting (the order of components is of a tuple is random):

$$p_{x,i} = [ue_\gamma(f_{x,A,i})\ ue_\chi(f_{x,X,i})\ ue_y(c_{\pi(i)})\ ue_y(c'_{\pi(i)})\ ue_y(id)\ ue_y(id')]$$

(γ and χ denote public keys of RS, $y = y_1 \cdot \ldots \cdot y_k$ is the joint public key of tallying authorities).

A 6-tuples $ap_{x,i}$ used for revocation have the following form (again, the components indicated are permuted at random):

$$[ue_\gamma(f_{x,A,i})\ ue_\chi(f_{x,X,i})\ ue_y(c_{\pi(K+i)})\ ue_y(c'_{\pi(K+i)})\ ue_y(rid)\ ue_y(rid')]$$

A ballot If a voter has chosen t (upper/lower) part of the row j, then a voting ballot has the form $v = [p_{t,j}]$, and the anti-vote contains: $av = [ap_{t,j}; p_{t,j}]$.

4.2 Voting Process

Part I: Ballot Generation Procedure

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

1. Alice requests $n \geq 2$ cards.
2. APP sends a request for n cards to BGS (may be through an anonymous channel),
3. BGS responds for each request with the following data :
   - voting cards,
   - commitments to the identifiers contained in these cards,
   - tokens $A, \overline{A}$,
   - commitments to the components s_1, s_2 of the cyclic shifts s used.
   - digital signature under all these data.
4. Alice obtains information about the cyclic shifts (s) and components of cyclic shifts (s_1, s_2) used in cards, T_A and the identifier id. Those information is sent through a channel that is inaccessible to the PC running APP (e.g. phone, SMS, ...).
5. Alice chooses a card i for voting.
6. Alice together with APP generates the deterministic signature under publicly known value z.
7. APP sends signature $\text{sign}_{Alice}(z)$ - this signature determines uniquely which shift component should by revealed,
8. BGS sends RSA-RE signatures to the card chosen for voting, data for opening the ciphertexts in other cards and checking the corresponding commitments, the shift component from the voting card indicated by $\text{sign}_{Alice}(z)$ and data for checking its commitment.
9. APP checks if received values satisfy commitments and displays to Alice the plaintexts and the requested shift component of the card chosen for voting.

Part II: Vote casting

1. Alice makes her choice: she chooses a part t (upper/lower) of a row w from the card, according to the cyclic shift used and her voting preferences. Namely, if she chooses the jth candidate, then the row w has to contain a ciphertext of c_j in the voting card, that is $\pi(w) = j$.
2. APP performs the following steps:
   (2.1) it creates a ballot by selecting $v = [p_{t,w}]$ and revoking-ballot $av = [ap_{t,w}; p_{t,w}]$,
   (2.2) APP modifies the values contained in v and av by re-encrypting every ciphertext contained in it.
   (2.3) APP contacts RS and obtains a challenge r.
   (2.4) Alice signs the challenge obtaining $\text{sign}_{Alice}(r)$, a deterministic signature scheme is used.
   (2.5) APP derives (in a deterministic way) an encryption key $K := \mathbf{R}(\text{sign}_{Alice}(r))$ with a pseudorandom generator $\mathbf{R}$.
   (2.6) Alice signs v and $Enc_K(av)$; then APP sends these data together with the signatures created to RS.
3. RS checks the signatures and the correctness of v. Then v is stored in the set of votes cast on the public Bulletin Board, the second (encrypted) packet is stored in a repository of revocation codes. RS also provides a receipt for Alice which is a signature of RS under both packets.
4. Alice checks, with the help of Bulletin Board, if $ue_\gamma(f_{t,A,i})$ sent with her vote agrees with the value indicated by T_A known to her.

Part III: 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 Bulletin Board and increments the number of the canceled votes.

Before accepting a revocation vote, RS checks if plaintexts of $ue_\chi(f_{t,X,i})$ contained in vote v and revoking-vote av are the same (this prohibits canceling a vote for a different candidate).

Part IV: Tallying process After closing the polling stations the RS servers are closed as well. From every vote v and revoking-vote av, RS takes the ciphertexts containing voting options and identifiers and sends them to the first tallying authority.

Now, the mixing procedure is executed by an array of mix servers run by independent tallying authorities. For $1 \leq i \leq \lambda$, the ith tallying authority runs a server that executes the following steps:

– it reads the RSA-RE ciphertexts from Bulletin Board $i - 1$ and checks the

signatures of BGS,

– it partially decodes the ciphertexts in each block with its private key,

– it re-encrypts each ciphertext,

– it permutes the ciphertexts at random and posts them on Bulletin Board i.

The last tallying authority gets, after decryption, plaintexts of the ciphertexts included in the ballots. It presents them in the Bulletin Board λ together with a Zero Knowledge Proof of correct decoding. Now, on Bulletin Board λ one can see election results.

Part V: Vote Counting It is checked that there are exactly two occurrences of each identifier r. Then, one counts number of votes for every candidate, subtracts the number of anti-votes and divides by 2.

5 Final Remarks

Communication Complexity A voting card contains N positions of candidates or voting options, so there are N parts (upper/lower, vote/anti-vote). Each of those parts contains 4-tuples $(id, id', c_i, c_i^{'})$ of RSA-RE-onions and two encrypted marks $A/\overline{A}, X/\overline{X}$, hence together 18 ciphertexts. There are $4K$ parts, so there are about $100N$ ciphertexts. If 1024-bit encryption is used, then the total size of the card is about $10N$ kB, so for $K = 20$ it is about 200 kB.

Social Aspects 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. As far as we know this is the only solution so far that works without any trusted element. The main weak point is quite complex design incomprehensible for an average voter, so social acceptance might be a problem.

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.: Punch Scan, Workshop on Frontiers in Electronic Elections 2005, `http://www.punchscan.org`
4. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84-88, 1981.
5. Chaum, D., Ryan, P. Y. A., Schneider, S.: A Practical Voter-Verifiable Election Scheme. ESORICS '2005, LNCS 3679, 118-139.
6. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. Advances in Cryptology – CRYPTO '86, LNCS 263, 186-194.
7. Furukawa, J., Sako, K.: An Efficient Scheme for Proving a Shuffle. Advances in Cryptology-CRYPTO '2001, LNCS 2139, 368-387.

8. Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: Rapid Mixing and Security of Chaum's Visual Electronic Voting. ESORICS'2004, LNCS 2808, 132-145.
9. Golle, P.: Reputable Mix Networks. Privacy Enhancing Technologies (PET) 2004, LNCS 3424, 51-62.
10. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-encryption for Mixnets. CT-RSA '2004, 163-178.
11. Juels, A., Catalano, D., Jakobsson, M.: Coercion-Resistant Electronic Elections. WPES '2005, 61-70
12. Hirth, M.: Receipt-Free K-out-of-L Voting based on ElGamal Encryption. Workshop on Frontiers in Electronic Elections 2005.
13. Jakobsson, M.: A Practical Mix. Advances in Cryptology- EUROCRYPT '1998, LNCS 1403, 448-461.
14. Jakobsson, M.: Flash Mixing. ACM Symposium on Principles of Distributed Computing '1999, 83-89.
15. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. USENIX Security Symposium '2002, 339-353.
16. Karlof, C., Sastry, N., Wagner, D.: Cryptographic Voting Protocols: a Systems Perspective. USENIX Security Symposium '2005, 33–50.
17. Klonowski, M., Kutyłowski, M., Lauks, A., Zagórski, F.: Universal Re-encryption of Signatures and Controlling Anonymous Information Flow. Wartacrypt 2004.
18. Klonowski, M., Kutyłowski, M., Lauks, A., Zagórski, F.: A Practical Voting Scheme with Receipts. International Security Conference (ISC)'2005, LNCS 3650, 380-393.
19. Lee, B., Kim, K.: Receipt-Free Electronic Voting Scheme with a Tamper-Resistant Randomizer. Information Security and Cryptology - ICISC 2002, LNCS 2587, 389-406.
20. Mitomo, M., Kurosawa, K.: Attack for Flash MIX. Advances in Cryptology- ASIACRYPT '2000, LNCS 1976, 192-204.
21. McGaley, M.: Report on DIMACS Workshop on Electronic Voting - Theory and Practice, http://dimacs.rutgers.edu/SpecialYears/2003_CSIP/reports.html
22. Moran, T, Naor M.: Receipt-Free Universally-Verifiable Voting with Everlasting Privacy, Advances in Cryptology- CRYPTO '2006, LNCS 4117, 373-392
23. Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting. ACM Conference on Computer and Communications Security '2001, 116-125.
24. Rivest, L. R.: The Three Ballot voting system. http://theory.csail.mit.edu/~rivest/Rivest-TheThreeBallotVotingSystem.pdf
25. Rivest, L.R.: voting resources page, http://theory.scail.mit.edu/~rivest/voting/
26. Smith, W. D.: Cryptography Meets Voting. http://www.math.temple.edu/~wds/homepage/cryptovot.pdf

6 Appendix

Elections →	traditional	electronic [18]	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

Fig. 1. A comparison of voting techniques