A Cryptographic Scheme for Computerized General Elections

Kenneth R. Iversen
Department of Electrical Engineering and Computer Science
Norwegian Institute of Technology
7034 Trondheim, Norway
kenneth.iversen@idt.unit.no

Abstract

This paper presents a novel cryptographic scheme which fully conforms to the requirements of holding large scale general elections. The participants of the scheme are the voters, the candidates and the government. The scheme ensures independence between the voters in that they do not have to be present at the same time or go through several phases together; no global computation is needed. The scheme preserves the privacy of the votes against any subset of dishonest voters, and against any proper subset of dishonest candidates, including the government. Robustness is ensured in that no subset of voters can corrupt or disrupt the election. This also means that no voter is able to vote more than once without being detected. The verifiability of the scheme ensures that the government and the candidates cannot present a false tally without being caught. "Voting by telephone" is possible by employing the proposed scheme.

1 Introduction

This paper presents a cryptographic scheme for secret ballot elections. It is a scheme which fully conforms to the requirements of holding large scale general elections. The scheme involves the eligible voters, the government, and the candidates the voters can vote on. The basic assumptions of the scheme are that each voter can communicate all the candidates (from now on, this includes the government) simultaneously, and that at least one candidate do not collaborate with the others. Under these assumptions, the scheme is robust in that no subset of dishonest voters and no proper subset of dishonest candidates can disrupt or corrupt the election, and the privacy of the votes and voters is preserved.

The verifiability of the scheme is restricted to the candidates. (It is possible to include other, possibly more trustworthy parties.) Assuming that every voter trusts one of these parties, this yields public verifiability. The verifiability ensures, with overwhelming probability, that the government cannot present a false tally without being caught.

The scheme is well suited for implementing a "voting by telephone" scheme.

© Springer-Verlag Berlin Heidelberg 1992

1.1 Relation to Previous Work

There have been several publications on the problem of holding elections by employing computers and cryptographic protocols, and several cryptographic schemes have been proposed where the voters openly send encrypted messages back and forth until they all are confident of the outcome of the election (boardroom voting) [DLM82, Yao82, Mer83].

The problems with these schemes are that one has to know in advance who wish to vote, and if any voter stops following the protocol during the election, the election cannot be completed. Such schemes are clearly not suitable for real-world elections.

Chaum has given an election scheme that makes use of one or several trusted "mixes" to scramble pairs of votes and digital pseudonyms [Cha81].

Whereas Chaum's scheme hides the identity of the voters, the scheme of (Cohen) Benaloh *et al.* hides the actual value of the vote [CF85, BY86, Ben87]. They take a quite different approach, by employing the hardness of deciding higher residues and interactive protocols.

My work has been much inspired by that of Benaloh *et al.*, and it has adopted many of their ideas. The scheme of this paper does to some extent conform to their election paradigm. The major problem that their scheme succumbs to is that it requires the participants to go through several phases together, where one phase cannot start before all the participants have finished the previous. This problem is solved in my scheme; all voters register and vote totally independent.

The scheme presented is a very practical and flexible election scheme. It can be used to implement almost any election setting; from the conventional setting where each voter show up (independently!) at the voting place, register, vote, and goes back home; to some kind of "voting by telephone" setting.

Chaum has given another method of holding verifiable secret ballot elections that removes the need for a mix [Cha88]. The work is similar to that of a boardroom election in that a failure of a single voter can disrupt the election. However, Chaum's method ensures that such failures can be traced. This allows an election to be restarted without the faulty voter, but this approach is not practical for large-scale elections.

Boyd has proposed a voting scheme based on the use of "multiple key ciphers" [Boy88, Boy90]. It ensures that votes cannot be forged, and privacy is preserved, provided the voters can deliver their votes anonymously. The major problem of this scheme is that the government can see the votes delivered and even worse, produce a false tally by adding votes of its own choice; there is no verifiability.

2 The Election Privacy Homomorphism

I here present the privacy homomorphism that is used to construct the ballots of the election scheme. The privacy homomorphism is additive and probabilistic, i.e., the cleartext domain operation is (modular) addition and there are several different and uncorrelated encryptions of the same number. For full details, see Ref. [Ive91a].

2.1 Election Triples

Definition 2.1 Let k be a security parameter, e be a fixed small prime, p be a $k/2$-bit prime such that $e|(p-1)$, and q be a $k/2$-bit prime such that $e \not| (q-1)$. Further, let $n = pq$. Finally, let g be an element in $\mathbf{Z}_n^*$ such that e divides the order of g. For such e, g, and n, I define an *election triple* to be (e, g, n).

I will throughout let G be the set of powers of g modulo n; $G = \{g^j \pmod n) | j \geq 1\}$.

2.2 Index Classes

Definition 2.2 Let (e, g, n) be an election triple. Let $w \equiv g^v \pmod n \in G$, for some integer v. The *index class* of w, denoted $[\![w]\!]_{(e,g,n)}$ (or simply $[\![w]\!]$ when (e, g, n) is given), is $v \pmod e$. If $w \notin G$, I say that $[\![w]\!]$ is undefined.

Definition 2.3 An election triple (e, g, n) is said to be *valid* when e divides the order of g. The election triple is said to be *good* if g in addition is a generator modulo p.

In Ref. [Ive91b] it is devised an efficient perfect zero-knowledge protocol that enables the publisher of an election triple to convince anyone who wants to be assured that the triple is valid, without giving away any information about the secrets involved.

I am now ready to describe how to use the privacy homomorphism. Given a good election triple (e, g, n), the values to be encrypted must be in the set $\mathbf{Z}_e$.

How to Encrypt

Suppose a party A wants to encrypt a number $v \in \mathbf{Z}_e$. Then,

> A chooses $r \in_R \mathbf{Z}_n$, and computes $x = v + re$,
>
> A computes $E(v) \equiv g^x \pmod{n}$.

In general, $\lfloor \log_2 e \rfloor$ bits of cleartext is expanded into k bits of ciphertext. Since there are several different encryptions of the same value, test for equality is not possible.

2.3 The Election Privacy Homomorphism Assumption

In this section, I formally state the intractability assumption for the problem of deciding index classes in the election privacy homomorphism. Clearly, the problem cannot be harder than factoring or computing discrete logarithms modulo a composite integer (see open problems 5 and 22 in Ref. [AM87]). No efficient algorithm for solving the problem without knowing the factorization of the modulus is known.

In order to state the intractability assumption, I introduce the predicate $IND_{(e,g,n)}$. For all $w \in G$,

$$IND_{(e,g,n)}(v, w) = \begin{cases} 1 & \text{if } [w] = v \\ 0 & \text{if } [w] \neq v \end{cases} .$$

Assumption 2.1 (EPH Assumption (EPHA)) *Let $v \in \mathbf{Z}_e$. Then for all polynomial size families of circuits $C = \{C_k\}_{k \geq 1}$, with $(3k + |e|)$-bit input gates, for any good election triples (e, g, n) such that $|n| = k$, and for all $w \in G$,*

$$\Pr(C_k(v, w, g, n) = \mathrm{IND}_{(e,g,n)}(v, w)) < \frac{e-1}{e} + \nu(k),$$

where the probability is taken over the random inputs of C_k. The fraction $\frac{e-1}{e}$ is the probability of guessing correctly if C_k always outputs 0. $\nu(k)$ is a function that vanishes faster than the inverse of any polynomial in k.

In Ref. [Ive91a], I show, in a manner similar to that of Goldwasser and Micali [GM84], that the privacy homomorphism is a probabilistic public key encryption function based on the above assumption. I further show that the problem of computing index classes is "everywhere hard".

3 EPH Based Votes

I will now describe what the ballots and votes used in the election scheme will look like.

Before the election starts, I assume that every candidate has published a (preferably good, but possibly valid) election triple. Let (e, g_i, n_i) be the election triple of candidate i, $1 \leq i \leq \sigma$. e must be larger than the number of eligible voters.

Definition 3.1 Let (e, g_i, n_i) be the election triple of candidate i, $1 \leq i \leq \sigma$. A *vote* w is a σ-tuple $w = (g_1^{v_1} \pmod{n_1}, \ldots, g_\sigma^{v_\sigma} \pmod{n_\sigma})$.

Definition 3.2 A *ballot* W is a σ-tuple $W = (w_1, \ldots, w_\sigma)$ of votes.

Definition 3.3 *The* index class tuple *(or just index tuple) of a vote $w = (g_1^{v_1}, \ldots, g_\sigma^{v_\sigma})$ is the tuple $(v_1 \pmod{e}, \ldots, v_\sigma \pmod{e})$. I denote it $⟦w⟧$.*

Further, for a vote $w = (g_1^{v_1}, \ldots, g_\sigma^{v_\sigma})$, meaning $v = \sum_{i=1}^{\sigma} v_i \pmod{e}$, I will write $v = \sum ⟦w⟧$.

Definition 3.4 A vote $w = (g_1^{v_1}, \ldots, g_\sigma^{v_\sigma})$ is *valid* if $v = \sum ⟦w⟧ = 0$ or 1. A vote where $v = 0$ is called a *no-vote* and a vote where $v = 1$ is called a *yes-vote*.

Definition 3.5 A ballot $W = (w_1, \ldots, w_\sigma)$ is *valid* if every vote $w_1, \ldots, w_\sigma$ are valid and $\sum_{j=1}^{\sigma} v_j = \sum_{j=1}^{\sigma} \sum ⟦w_j⟧ = 1 \pmod{e}$.

I will refer to the $v_j = \sum ⟦w_j⟧$ as the *actual vote* for candidate j.

Now, instead of the candidates having to store each ballot from all the voters, the homomorphism property comes into use. Let W_1 and W_2 be two valid ballots. It should not be hard to see that to store the sum of the actual votes one can store the componentwise product of the votes in W_1 and W_2.

Let the final net ballot be $\widehat{W} = (\widehat{w}_1, \ldots, \widehat{w}_\sigma)$. Then the final number of yes-votes for candidate j is $\sum ⟦\widehat{w}_j⟧$.

4 Unreusable Eligibility Tokens

The basic assumptions of the election scheme will be applied here also; the voter communicates with all the candidates simultaneously, and at least one candidate is honest. Let the number of candidates be σ.

The scheme for providing unreusable eligibility tokens is a modification to the scheme for providing unreusable electronic cash presented in Ref. [CFN90].

4.1 Initialization

The computations and actions described below can be done at any time before the process of token issuing starts, but only in the order indicated by the numbering.

1. The candidates agree on and publish two (even) security parameters k and s, a public one-way collision-free hash function h, and a secure public digital signature scheme ([GMR88]) to be used by the voters. Each candidate j then publishes its public RSA key (e_j, n_j) (the corresponding secret key is d_j), such that $|n_j| = s$.
2. For $i = 1, \ldots, k$, the voter chooses integers a_i, b_i, t_i, r_i, and $z_i \in_R Z_n^*$, where $n = \max_{j=1,\ldots,\sigma}(n_j)$, and computes the inverse of r_i modulo each of the candidates' RSA modulus. The voter then prepares a digital signature on $h(z_1)||h(z_2)||\ldots||h(z_k)$. Let S_V denote this signature.

4.2 Token Issuing

Some time before the election day, the voter presents and identifies him- or herself to the candidates (in an election office handling eligibility), and gives his or her public signature key to the candidates. The candidates create a string ID_V which contains the voter's name, ID number, or any other information that the candidates want to establish. The voter and the candidates then perform the protocol below.

1. For $i = 1, \ldots, k$, the voter computes the blinded values

$$v_i = \{r_i^{e_j} \cdot h(h(a_i||b_i)||h(a_i \oplus (ID_V||z_i)||t_i)) \pmod{n_j}\}_{j=1,\ldots,\sigma}.$$

   The voter sends $\{v_i\}_{i=1,\ldots,k}$ to the candidates. In addition, the voter supplies the candidates with the digital signature on $h(z_1)||h(z_2)||\ldots||h(z_k)$; S_V.

2. The candidates perform a sub-protocol and send to the voter a random subset of $k/2$ distinct indices $I = \{i_j\}_{j=1,\dots,k/2}$, where for all $j, 1 \le i_j \le k$.

3. For all $i \in I$, the voter reveals a_i, b_i, t_i, r_i, and z_i to the candidates.

4. For all $i \in I$, the candidates check that the voter computed the v_i correctly in Step 1. In addition, the candidates check that $h(z_i)$ is among hash values signed by the voter. If any of the candidates discover any fallacies they terminate the protocol.

5. For simplicity, let the remaining indices not in I be $1, \dots, k/2$. Each of the candidates computes and sends the RSA signature, S_{C_j}, of the $k/2$ unopened values to the voter.

$$\{S_{C_j} = \prod_{i=1}^{k/2} r_i \cdot h(h(a_i \| b_i) \| h(a_i \oplus (ID_V \| z_i)) \| t_i))^{d_j} \pmod{n_j}\}_{j=1,\dots,\sigma}.$$

6. The voter now removes the blinding and extracts the unreusable eligibility token

$$ET = \{\prod_{i=1}^{k/2} h(h(a_i \| b_i) \| h(a_i \oplus (ID_V \| z_i)) \| t_i))^{d_j} \pmod{n_j}\}_{j=1,\dots,\sigma}.$$

After executing the initialization protocol, the candidates store ID_V, S_V, and $\{z_i\}_{i \in I}$. During the initialization protocol, the candidates have verified that each of the $k/2$ v_i's they examined generates an appropriate $ID_V \| z_i$. I will now assume that the candidates have legal proof that the voter has voted more than once if they can present the preimage of at least $(k/2) + 1$ of the hash values $h(z_i)$ in S_V.

4.3 Using the Token

On the election day, when the voter is using the token in the voting process, the protocol below is performed by the voter and the candidates.

1. The voter sends ET to the candidates.

2. The candidates perform a sub-protocol to obtain the challenge string c, and send it to the voter. $c = \{c_i \in_R \{0, 1\}\}_{i=1,\dots,k/2}$.

3. For $i = 1, \dots, k/2$, the voter sends the values y_i to the candidates.

$$y_i = \begin{cases} a_i, b_i, h(a_i \oplus (ID_V \| z_i)) \| t_i) & \text{if } c_i = 0 \\ h(a_i \| b_i), a_i \oplus (ID_V \| z_i), t_i & \text{if } c_i = 1 \end{cases}$$

4. The candidates check that the y_i's fit ET.

   If any of the checks fails, the candidates halt and reject, otherwise they halt and accept.

When the protocol is finished with the candidates accepting, the candidates check whether the token has been used before, by searching in a database where all the previously received tokens are stored. If it is not used before, the candidates store ET, the challenge string $\{c_i \in_R \{0,1\}\}_{i=1,\dots,k/2}$, and the values a_i, if $c_i = 0$, and $a_i \oplus (ID_V \| z_i)$, if $c_i = 1$. If the candidates discover that the token has been used before, then with overwhelming probability, any candidate is able to extract the identity ID_V of the voter, and provide a legal proof of the fact that the voter has voted twice.

4.4 ET Security

The security of the unreusable electronic cash scheme was left as an open challenge in Ref. [CFN90], and no attempts to solve the problem have been made here. Ref. [Ive91b] gives a proof of unreusability.

5 The Election Scheme

The participants of the scheme are the eligible voters and the candidates. Let the number of eligible voters and candidates be ρ and σ, respectively, such that $\sigma < \rho$. Note that the registration and voting phases can be performed independently by each voter.

5.1 Election Initialization

The candidates do what is described in Step 1 in Section 4.1. They then execute the following *election initialization protocol*:

Agreeing on e: The candidates agree on a prime e which is larger than the number of eligible voters ($e > \rho > \sigma$).

Generating election triples: For $i = 1$ to σ, candidate i secretly produces two random $s/2$-bit primes p_i and q_i, such that $e|(p_i - 1)$ and $e \not| (q_i - 1)$. Let $n_i = p_i q_i$. Candidate i also chooses an element $g_i \in \mathbf{Z}_{n_i}^*$ which is a generator modulo p. p_i and q_i are kept secret, while (e, g_i, n_i) is published as candidate i's election triple.

Each candidate in turn must then give a zero-knowledge proof to show that the election triple is valid to any candidate who wants to be assured.

In the sequel of this chapter, all computations are done modulo the n_i's. Which applies where should be clear from the subscript of the g_i involved.

Finally the candidates compute an initial "net ballot" $\widehat{W}_0 = (\widehat{w}_{0,1}, \ldots, \widehat{w}_{0,\sigma})$, such that for all j, $\sum (\![\widehat{w}_{0,j}]\!) = 0$. Each of the candidates then signs a copy of the hash value $h(\widehat{W}_0 \| 0)$ using the secure digital signature scheme, and then publishes it. h is employed for efficiency reasons only. The zero that is concatenated with the ballot is the (initial) sequence number. This signing is to avoid that, when the election is finished, any proper subset of dishonest candidates can construct their own final ballot and claim it to be the real one.

5.2 Voter Registration

An eligible voter first performs the eligibility token initialization described in Step 2 in Section 4.1. After this, she appears and identifies herself at a registration office to obtain an unreusable eligibility token, ET, produced by the protocol given in Section 4.2 with security parameter $2k$.

5.3 Voter Initialization

At some time before the actual voting is to take place, the voter decides which candidate she wants to give a vote to. A vote on candidate 1 (the government) might yield a blank vote. I will for simplicity assume that the voter votes blank.

The voter then performs the initializing computations shown below. Note that these computations can be done off-line.

1. The voter prepares the ballot W according to the following program:

```
FOR $j = 1$ TO $\sigma$ DO
  FOR $i = 1$ TO $\sigma - 1$ DO $v_{i,j}$ := a random element in $\mathbf{Z}_{n_i}$
  $v_{\sigma,j}$ := an element in $\mathbf{Z}_{n_\sigma}$ s.t. $\sum_{i=1}^{\sigma} v_{i,j}$ = IF $j = 1$ THEN 1 ELSE 0
  $w_j$ := $(g_1^{v_{1,j}}, \ldots, g_\sigma^{v_{\sigma,j}})$
END DO
```

```
$W$ := $((w_1, \ldots, w_\sigma), ET)$
```

2. The voter prepares the tuple $V = (v_1, \ldots, v_\sigma)$, where $v_i = \sum_{j=1}^{\sigma} v_{i,j}$.

3. For each vote w_j in the ballot the voter prepares k "test-pairs" according to the following program: (for simplicity, I drop the subscript j)

```
FOR $i = 1$ TO $k$ DO
  $bit_i$ := a random element in $\{0,1\}$
  FOR $j = 1$ TO $\sigma - 1$ DO
    $\alpha_{i,j}$ := a random element in $\mathbf{Z}_{n_i}$
    $\beta_{i,j}$ := a random element in $\mathbf{Z}_{n_i}$
  END DO
  $\alpha_{i,\sigma}$ := an element in $\mathbf{Z}_{n_\sigma}$ s.t. $\sum_{i=1}^{\sigma} \alpha_{i,j} = 0$
  $\beta_{i,\sigma}$ := an element in $\mathbf{Z}_{n_\sigma}$ s.t. $\sum_{i=1}^{\sigma} \beta_{i,j} = 1$
  $a_i$ := $(g_1^{\alpha_{i,1}}, \ldots, g_\sigma^{\alpha_{i,\sigma}})$
  $b_i$ := $(g_1^{\beta_{i,1}}, \ldots, g_\sigma^{\beta_{i,\sigma}})$
  $pair_i$ := IF $bit_i = 0$ THEN $(a_i, b_i)$ ELSE $(b_i, a_i)$
END DO
$PAIR$ := $\{(pair_{1,j}, \ldots, pair_{k,j})\}_{j=1,\ldots,\sigma}$
```

5.4 Voting

When voting, the voter performs the protocol below with the candidates.

Repeat Steps 1—4 k times ($i = 1, \ldots, k$) (for each vote w_j in parallel). (For simplicity, I drop the subscript j.)

1. If $i = 1$, the voter sends the vote w to the candidates. The voter sends $pair_i$ to the candidates.

2. The candidates perform a sub-protocol to obtain a random challenge bit c_i, and send it to the voter.

3. The voter answers with d_i.

   If $c_i = 0$ then $d_i = ((\alpha_{1,i}, \ldots, \alpha_{\sigma,i}), (\beta_{1,i}, \ldots, \beta_{\sigma,i}))$. If $c_i = 1$ then $d_i = (v_1 + \alpha_{1,i}, \ldots, v_\sigma + \alpha_{\sigma,i})$ if $\sum [\![w]\!] = 1$ and $d_i = (v_1 + \beta_{1,i}, \ldots, v_\sigma + \beta_{\sigma,i})$ if $\sum [\![w]\!] = 0$.

4. If $c_i = 0$, the candidates check that $pair_i[1] = (g_1^{d_i[1,1]}, \ldots, g_\sigma^{d_i[1,\sigma]})$ and $pair_i[2] = (g_1^{d_i[2,1]}, \ldots, g_\sigma^{d_i[2,\sigma]})$, or possibly vice versa. If $c_i = 1$, the candidates check that $\sum_{j=1}^{\sigma} d_i[j] = 1$ and that $(g_1^{d_i[1]} \ldots, g_\sigma^{d_i[\sigma]}) = w\cdot pair[1]$ or $w\cdot pair[2]$. If any candidate discovers any errors, the candidates halt and the voter is excluded from the election.

5. Finally, when steps 1—4 have been repeated k times, the voter sends V to the candidates.

6. The candidates check that $\sum_{i=1}^{\sigma} v_i = 1$ and that, for each i, $g^{v_i} = \prod_{j=1}^{\sigma} w_j[i]$.

Besides the voting protocol the voter and the candidates perform the token usage protocol described in Section 4.3. This can easily be embedded in the voting protocol.

If none of the candidates have discovered any fallacies in the voting protocol or the token usage protocol, they accept the ballot, and indicates this to the voter by sending him or her signed "receipts" (of some sort). The candidates then compute the net ballot $\widehat{W}_m = (\widehat{w}_{m-1,1} \cdot w_{m,1}, \ldots, \widehat{w}_{m-1,\sigma} \cdot w_{m,\sigma})$, where $w_{m,j}$ is vote j of voter m. Again, each candidate signs a copy of $h(\widehat{W}_m \| m)$, as described in the election initialization protocol.

5.5 Tally Computing

When the election is finished the final net ballot is $\widehat{W}_{\rho'} = (\widehat{w}_{\rho',1}, \ldots, \widehat{w}_{\rho',\sigma})$, where ρ' is the number of voters that actually voted during the election. Now, the total number of yes-votes cast for candidate j is $\sum [\![\widehat{w}_{\rho',j}]\!]$. To be able to compute this tally, the candidates have to publish their "sub-tallies", i.e., candidate i publishes the tuple $([\![\widehat{w}_{\rho',1}[i]]\!], \ldots, [\![\widehat{w}_{\rho',\sigma}[i]]\!])$, and so forth. They must in addition give a (perfect) zero-knowledge proof of the validity of the published "sub-tallies". See full paper for reference.

6 Security

The first thing to notice is that the voting protocol is σ versions of a computational zero-knowledge protocol, given in Ref. [Ive91b]. Note also that the protocol for each vote in the ballot is run sequentially, so the protocol is still zero-knowledge.

Theorem 6.1 (Completeness) *The ballot of an honest voter is accepted by honest candidates with probability one.*

Proof: The fact that each valid vote is accepted with probability one follows directly from the completeness part of the proof of zero-knowledgeness of the voting protocol (see Ref. [Ive91b]). In addition, for the whole valid ballot, the check performed by the candidates in Step 6 in the voting protocol will always be accepted. ▮

Theorem 6.2 (Soundness) *If at least one candidate is honest, then, with overwhelming probability, a dishonest voter will not succeed in delivering an invalid ballot.*

Proof: That this holds for each of the votes in the ballot follows directly from the soundness part of the proof of zero-knowledgeness of the voting protocol (see Ref. [Ive91b]). In addition, the fact that $\sigma < e$ implies that if more than one of the votes are yes-votes, then $\sum_{i=1}^{\sigma} v_i > 1$, and the honest candidates will not accept. ▮

Theorem 6.3 (Privacy) *Under the EPHA, if at least one candidate is honest, the privacy of the votes is preserved.*

Proof: The voting protocol is proven to be computational zero-knowledge (see Ref. [Ive91b]), and this implies that no information about the value of the votes can be extracted only from executing the protocol. Let Γ denote any subset of dishonest candidates such that $|\Gamma| < \sigma$. Let, for simplicity, the candidates in Γ be $C_1, C_2, \ldots, C_{\sigma-1}$, and thus $|\Gamma| = \sigma - 1$. First, the candidates in Γ cannot extract any information from the index class of the elements they are able to decrypt, i.e., from $[\![w[i]]\!], i < \sigma$ in any votes. The other element $w[\sigma]$, will to the candidates in Γ, be a random element in G_σ.[1]

From the above it follows that a polynomial advantage in determining the actual vote for some candidate in some ballot (delivered by an honest candidate) yields a polynomial advantage in determining the actual vote for any candidate in any ballot.

Assume now that the candidates in Γ have gained such a polynomial advantage (somehow). But, then this is a polynomial advantage in determining the index class of at least one element $w[\sigma]$, in any vote in any ballot. This clearly contradicts the EPHA, and our assumption must be wrong. ▮

Theorem 6.4 (Unreusability) *With overwhelming probability, no voter is able to vote more than once without being detected by an honest candidate.*

[1] Recall that $G_i = \{g_i^j \pmod{n_i} \mid j \geq 1\}$, where (e, g_i, n_i) is the election triple published by candidate i.

Proof: This follows from the proof of unreusability of the eligibility tokens. See Ref. [Ive91b]. ▮

Theorem 6.5 (Tally correctness) *Under the assumptions that the employed signature scheme is secure and that at least one candidate is honest, then, with overwhelming probability, the published tally is equal to the actual result of the election.*

Proof: To be able to claim the validity of a published tally, the (claimed) final ballot must be shown together with signed copies from all the candidates. If this is the case, the properties of the privacy homomorphism ensures that at most one tally can be produced from this final ballot (see Ref. [Ive91a]). By the proof of soundness, every ballot "in" the published net ballot is valid with overwhelming probability, and thus *exactly* one tally can be produced from it.

No proper subset of dishonest candidates can produce a valid final ballot with an equal or larger sequence number without breaking the signature scheme. ▮

Theorem 6.6 (Eligibility) *Under the assumption that the RSA blind signature scheme is secure, and if at least one candidate is honest, then with overwhelming probability, only eligible voters are able to deliver a ballot successfully.*

Proof (sketch): It is not known under what assumptions the theorem holds. See Section 4.4. ▮

The above election scheme enables voters to deliver their votes independent of each other. No subset of voters can disrupt the election, and the same applies to any subset of less than σ candidates.

7 Discussion

The scheme is very efficient in that nearly all time-consuming computations can be done offline. In the most time-critical protocol – the voting protocol, the voter need not do any time-consuming computations.

The election scheme presented here has one important drawback; the possibility for a voter to be paid to vote for a dishonest candidate, and afterwards be able to prove to this candidate that he or she actually did so.[2] It remains an open problem to fix this problem.

References

[AM87] L. Adleman and K. McCurley. Open Problems in Number Theoretic Complexity. In *Discrete Algorithms and Complexity*, pages 237–262. Academic Press, Inc., 1987.

[Ben87] J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, USA, September 1987. YALEU/DCS/TR-561.

[Boy88] C. Boyd. Some Applications of Multiple Key Ciphers. In C. G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88 Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 455–467. Springer-Verlag, 1988.

[Boy90] C. Boyd. A New Multiple Key Cipher and an Improved Voting Scheme. In J.-J. Quisquater, editor, *Advances in Cryptology - EUROCRYPT '89 Proceedings*, volume 434 of *Lecture Notes in Computer Science*, pages 617–625. Springer-Verlag, 1990.

[BY86] J. Benaloh and M. Yung. Distributing the Power of a Government to Enhance the Privacy of Voters. In *Proceedings of the 5th ACM Symposium on the Principles in Distributed Computing*, pages 52–62, 1986.

[CF85] J. Cohen and M. Fisher. A Robust and Verifiable Cryptographically Secure Election Scheme. In *Proceedings of the 26th Annual IEEE Symposium on the Foundations of Computer Science*, pages 372–382, 1985.

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88 Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, 1990.

[Cha81] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Cha88] D. Chaum. Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA. In C. G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88 Proceedings*, volume 330 of *Lecture Notes in Computer Science*, pages 177–182. Springer-Verlag, 1988.

[2]Thanks to Amir Herzberg for pointing this out.

[DLM82] R. DeMillo, N. Lynch, and M. Merritt. Cryptographic Protocols. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing*, pages 383–400, 1982.

[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and Systems Sciences*, 28(2):270–299, April 1984.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[Ive91a] K. Iversen. A Novel Probabilistic Additive Privacy Homomorphism. In *Proceedings of the International Conference on Finite Fields, Coding Theory, and Advances in Communications and Computing*, Lecture Notes in Pure and Applied Mathematics. Marcel Dekker, August 1991. To appear.

[Ive91b] K. Iversen. *The Application of Cryptographic Zero-Knowledge Techniques in Computerized Secret Ballot Election Schemes.* Doktor ingeniør-avhandling 1991:15, Norwegian Institute of Technology, February 1991. IDT-report 1991:3.

[Mer83] M. Merritt. *Cryptographic Protocols.* PhD thesis, Georgia Institute of Technology, USA, February 1983. GIT-ICS-83/6.

[Yao82] A. Yao. Protocols for Secure Computations. In *Proceedings of the 23rd Annual IEEE Symposium on the Foundations of Computer Science*, pages 160–164, 1982.