Self-tallying Elections and Perfect Ballot Secrecy

Aggelos Kiayias[1] and Moti Yung[2]

[1] Graduate Center, CUNY, NY USA,
akiayias@gc.cuny.edu
[2] CertCo, NY USA
moti@cs.columbia.edu

Abstract. Strong voter privacy, although an important property of an election scheme, is usually compromised in election protocol design in favor of other (desirable) properties. In this work we introduce a new election paradigm with strong voter privacy as its primary objective. Our paradigm is built around three useful properties of voting schemes we define: (1) *Perfect Ballot Secrecy*, ensures that knowledge about the partial tally of the ballots of any set of voters is *only* computable by the coalition of *all* the remaining voters (this property captures strong voter privacy as understood in real world elections). (2) *Self-tallying*, suggests that the post-ballot-casting phase is an open procedure that can be performed by any interested (casual) third party. Finally, (3) *Dispute-freeness*, suggests that disputes between active parties are prevented altogether, which is an important efficient integrity component.
We investigate conditions for the properties to exist, and their implications. We present a novel voting scheme which is the first system that is dispute-free, self-tallying and supports perfect ballot secrecy. Previously, any scheme which supports (or can be modified to support) perfect ballot secrecy suffers from at least one of the following two deficiencies: it involves voter-to-voter interactions and/or lacks fault tolerance (one faulty participant would fail the election). In contrast, our design paradigm obviates the need for voter-to-voter interaction (due to its dispute-freeness and publicly verifiable messages), and in addition our paradigm suggests a novel "corrective fault tolerant" mechanism. This mechanism neutralizes faults occurring before and after ballot casting, while self-tallying prevents further faults. Additionally, the mechanism is secrecy-preserving and "adaptive" in the sense that its cost is proportional to the number of faulty participants. As a result, our protocol is more efficient and robust than previous schemes that operate (or can be modified to operate) in the perfect ballot secrecy setting.

1 Introduction

One of the most challenging cryptographic protocol problems is electronic voting. We can distinguish two major settings for this problem: The first one based on homomorphic encryption was introduced in [CF85] (see also [BY86,Ben87]). The second one, based on anonymous channels (usually implemented through mixnets, e.g. [OKST97,Jak99]), was introduced in [Cha81]; for a related approach

D. Naccache and P. Paillier (Eds.): PKC 2002, LNCS 2274, pp. 141–158, 2002.

based on blind signatures see [Cha88,FOO92,Sak94]. Election protocols is a very active area of research which has progressively produced various tools for privacy and fault tolerance (e.g., [DLM82,Mer83,Yao82,Cha88,B89,PIK94,SK94,SK95], [BT94,CFSY96,CGS97,Abe99,Sch99]). There are, in fact, two notable sub-cases for the basic problem: (1) small scale (boardroom) elections, where the protocol is essentially run by the voters; (2) large scale (country wide) elections, where tally (or mix) authorities are involved. Typically, the first case allows for better privacy, while the second case implies the necessity of robustness where a voter misbehavior cannot disrupt the election.

Previous small scale voting-schemes, satisfy voter privacy in a strong way (e.g. small scale cases in [DLM82,Mer83,Yao82,Cha88,PW92]). However these schemes are very sensitive to failures, have high time/communication complexity requirements and/or require voter-to-voter private interaction (therefore are subject to costly disputes). On the other hand, large scale schemes rely on the assumption that (a subset of) the authorities do not try to violate the privacy of the voter by assuming some conditions. Examples of such conditions are: "at least one of the authorities remains honest (in mix-net based voting schemes, e.g. [Abe99])," or "at most t authorities collude against the voters but not more (distributed government schemes e.g. [BY86,CFSY96,CGS97])." The dependency of voter privacy on the authorities not colluding in such (large scale) schemes, was noted in [Bra99]. While the above achievements are remarkable, nevertheless, there is much more to wish for with respect to voter privacy in conjunction with efficiency, smooth operation and availability of election results.

Our main goal in this work is to investigate a new paradigm where the strong voter privacy of small-scale elections is possible, but to achieve it with increased efficiency and reliability (i.e., avoiding voter-to-voter interaction and tolerating faulty participants). The new paradigm will be able to combine strong voter privacy with the advantages of the bulletin board paradigm by separating a preprocessing step and an actual ballot casting step, while still implementing all communications via the bulletin board. Our contribution can be summarized as follows:

(1) On a Conceptual/Definitional Level
We introduce the concepts of Self-Tallying, Perfect Ballot Secrecy and Dispute-Freeness: A self-tallying voting-scheme, allows any casual third party to perform the tally computation and in general the post-ballot-casting phase (which becomes independent of any active participant's behavior and of any fault). Consequently, there are no authorities or voters involved in the post-election phase that becomes an *open procedure*. Furthermore, the ballots' availability implies that the counting can be split arbitrarily into chunks amongst a group of third-party talliers (without revealing any partial information about the votes) something that results in a speed-up of the tallying process. Perfect Ballot Secrecy (PBS) ensures that knowledge of the partial tally of the ballots of any set of voters (beyond what is known trivially) is only accessible to the coalition of the remaining voters (as secrecy is perceived in real world elections). Self-tallying and Perfect Ballot

Secrecy are both highly desired properties of election systems. As we will show, these properties are not consistent with traditional robustness of the large scale election schemes (started in [CF85] and followed by much of the homomorphic-encryption based work since then [BY86,Ben87,SK94,CFSY96,CGS97,Sch99], [FPS00,DJ00,KMO01]). Thus, we need a novel notion of fault tolerance which we suggest here: Corrective Fault Tolerance. It involves the honest users correcting faults which are publicly detectable due to Dispute-Freeness: a voting-scheme that is dispute-free has built-in prevention mechanisms that eliminate disputes between the active participants; in essence, any third party should be able to check whether an active participant follows the protocol. Such publicly verifiable scheme has a "public" audit trail which contributes to the reliability of the voting protocol (whereas, a typical small scale election lacks such reliability). Note that a dispute-free scheme cannot employ PKI or private channels between the participants.

(2) On a Protocol Design Level
We present a boardroom voting scheme which has a number of advantages compared to previous such voting schemes, both in efficiency and voter privacy. It is rooted in our new design paradigm which attempts to preserve as much reliability as possible, while employing homomorphic-encryption in the PBS setting. A key step in our design (that distinguishes it from past work) is that the voter does not select the randomness to be used in its ballot but rather it is generated distributively in a preprocessing stage ahead of ballot-casting. Our scheme is dispute-free, self-tallying and supports perfect ballot secrecy in the most efficient way: if all voters participate, the complexity of the post-ballot-casting phase is essentially optimal, requiring a total of at most $3n$ operations, where n is the number of ballots. Note that perfect ballot secrecy is achieved without voter-to-voter interaction. Our scheme can be viewed as optimizing the "all-honest" case (this design is known as the optimistic approach in the distributed computing literature). The protocol employs *corrective fault tolerance*, in the case of faulty disruption by participants. In such a fault tolerant scheme, the usual robustness properties of large scale schemes are replaced, since the latter are incompatible with perfect ballot secrecy. After ballot casting, no further faults are possible due to the employment of the notion of self-tallying. If PBS is not a prime objective our scheme can be readily transformed to a dispute-free, self-tallying, multi-authority based scheme that is more suitable for large scale elections.

Our voting paradigm allows batching many elections run among the same set of voters (boardroom setting) in such a way so that the online cost for the voter is essentially optimal (merely that of ballot-casting: constant in the number of participants). Additionally our paradigm allows the post-ballot-casting phase to be parallelized arbitrarily, breaking the tallying task into smaller tasks which can be assigned to casual third-party talliers that can work on problem instances independently. The combination of partial counts by talliers can be done in a straightforward manner without the voters and without loss of security. In contrast, if we adopt a typical homomorphic-encryption based scheme to the

case where "all voters are talliers" [BY86,Ben87,SK94,CFSY96,CGS97,Sch99], then, these schemes imply a post-ballot-casting involvement of the voters who must act as talliers, before the result can be announced. This is true also for the recent Paillier-based [Pai99] election schemes in [FPS00,DJ00].

2 Requirements for Voting Schemes

A voting-scheme needs to fulfill a variety of requirements to become useful. A brief presentation of requirements follows:

Privacy. Ensures the secrecy of the contents of ballots. Usually, in large scale elections, privacy is achieved by trusting some servers/ authorities. In small scale elections, privacy is achieved typically by intensive voter-to-voter communications.

Universal-Verifiability. Ensures that any party, including a casual observer, can be convinced that all valid votes have been included in the final tally.

Robustness. Ensures that the system can tolerate a certain number of faulty participants. Strong robustness implies that each voter is dealt with independently (which is the most suitable notion for large scale voting). This is shown to be impossible when all voters are also talliers (i.e., in a small scale election). As a result, in this work we consider a more relaxed form of robustness, a notion which we call **corrective fault tolerance**. Under this concept, fault tolerance is achieved through actions of the honest participants.

We note that the combination of universal-verifiability and robustness (which is seemingly contradicting) is a contribution of the fundamental work of Benaloh et al. [CF85,Ben87], to which the notion of enhanced privacy (via distributed talliers) was added in [BY86].

Receipt-Freeness. The voter cannot provide a receipt that reveals in which way he/she voted [BT94]. This was further developed in [SK95,Oka97,HS00] and it is a typical requirement in a country-wide elections to political offices.

Fairness. No partial-tally is revealed to anyone before the end of the election procedure [FOO92].

We introduce three additional desired properties for election schemes:

Dispute-Freeness. The fact that participants follow the protocol at any phase can be publicly verified by any casual third party. This property extends universal verifiability from the tallying phase to any part of the election protocol. A dispute-free voting scheme does not involve bilateral zero-knowledge proofs and dispute resolution mechanisms. In a dispute-free protocol the active **detection** of misbehaving is substituted with built-in **prevention** mechanisms which add integrity.

Self-Tallying. The post-ballot-casting phase can be performed by any interested (casual) third party. This property can be viewed as another strengthening of universal verifiability, in a different direction though. We note that self-tallying is a property inconsistent with the anonymous channel approach to elections, since in the post-ballot-casting phase a mix-network or some other mechanism between the active participants needs to be executed. Note also that a self-tallying scheme needs to cope with fairness explicitly, since the last voter to cast the last ballot may have access to the election results before choosing her vote. Nevertheless this can be dealt with effortlessly, as we will show, by having the authority that manages the deadline of the election cast a final "dummy vote" (in a publicly verifiable manner). Until this final vote, any partial tally is just a random value.

Perfect Ballot Secrecy. Ensures that knowledge of the partial tally of the ballots of a set of voters (beyond what is known and computed trivially by merely having the final tally) is *only* accessible to the coalition of *all* remaining voters. This property strengthens the privacy property and makes it independent of servers' behavior. Such high level of secrecy is a natural notion pertaining to elections and has appeared before in the context of non-robust procedures involving voter-to-voter communications in [Cha88].

2.1 Mutual Disjointness of Properties

We next point out that self-tallying and certain perfect ballot secrecy implementations are not consistent with strong robustness as the following propositions reveal. (The propositions are stated without a formal model, but can be instantiated formally as well; we avoid lengthy formalities since this subsection is merely a motivation for our design paradigm).

Proposition 1. *A self-tallying scheme cannot be robust and support privacy at the same time.*

To see this, assume that there is a scheme that is self-tallying and robust at the same time. Suppose that $n-1$ of n registered voters actually show up in the election. By the definition of the properties if the scheme is self-tallying and robust it should be possible for any third party (e.g., the timing authority) to compute the partial tally of the $n-1$ votes. In the case that all voters vote, this third party may perform the computation that can be performed when only $n-1$ voters show up therefore revealing the vote of the excluded voter, in violation of privacy.

Candidates for Perfect Ballot Secrecy (PBS) are voting-schemes based on distributing the power of talliers, where it is possible for the voters to simulate the computation of the distributed government. Alternative schemes involve using an anonymous channel that is implemented in such a way that no assumed trusted parties are employed, i.e. voter communication implements the anonymous channel (schemes where specific authorities are trusted not to collude, e.g.

[NSS91], are automatically excluded). Here we concentrate on the first approach as it appears to be more natural for obtaining efficient constructions in the PBS setting. Robustness in such schemes is usually based on some threshold secret-sharing mechanism (of messages or private keys), that produces shares among a set of authorities (this set, in fact, coincides with the set of voters in the PBS setting). In these schemes any set of t authorities may uncover the contents of any cast ballot. If the voters play the role of the authorities to achieve PBS, it is clear that the threshold t should be n, otherwise the PBS property is not satisfied. Obviously, setting the threshold to n implies lack of fault-tolerance. As a result we can conclude:

Proposition 2. *A voting-scheme with robustness based on secret sharing (of values or keys) cannot satisfy Perfect Ballot Secrecy.*

3 The Voting Scheme: Basic Steps and Mechanisms

The participants in the protocol are n voters denoted by $V_1, \ldots, V_n$ and the bulletin board authority. Each voter has a unique identification string denoted by $I(V_j)$. Identification strings are publicly known.

3.1 The Bulletin Board

A bulletin board is used for all necessary communication between the parties interacting in the voting scheme [CF85]. The bulletin board is a public-broadcast channel with memory. Any party (even third-parties) can read information from the bulletin board. Writing on the bulletin board by the active parties is done in the form of appending data in a specially designed area for each party. Erasing from the bulletin board is not possible, and appending is verified so that any third party can be sure of the communication transcript. In the sequel, the phrase "party X publishes value Y" means that X appends Y to the portion of the bulletin board that belongs to X. The bulletin board authority (server) participates in the protocol to alleviate the computational cost of the participants and administer the election. Server-based ciphertext processing helps in reducing the computations of the parties, whenever trusted. All computation performed by this authority will be publicly verifiable (e.g., by repeating the computation whenever not trusted). The bulletin board authority is also responsible for administering the election, namely, it performs actions such as starting and terminating the election, and maintaining a registry of the eligible voters that should gain access to the bulletin board.

3.2 Initialization

Let $\mathcal{G}_k$ be a family of groups, such that finding discrete logarithms in groups of $\mathcal{G}_k$ is hard (there is no probabilistic polynomial-time algorithm that finds discrete logarithms with non-negligible probability in k). For example, if p, q are

large primes with $q \mid p-1$, then the unique subgroup G of $\mathbf{Z}_p^*$ of size q, is an element of $\mathcal{G}_k$ where k is the number of bits of p, q. Let Gen be a probabilistic polynomial-time algorithm that given 1^k generates the description of a group $G \in \mathcal{G}_k$, and three random elements from G, f, g, h (with relative discrete logs unknown); this can be also be produced distributively, see [GJKR99]. We will denote the order of G by q. Arithmetic in the exponents is in $\mathbf{Z}_q$.

We assume that all parties, either observe Gen and its output, or are using a suitable cryptographic protocol for it. Therefore, all parties obtain the elements g, f, h. Every voter V_j selects a random value $\alpha_j \in \mathbf{Z}_q$ and publishes $h_j := h^{\alpha_j}$ (the voter's personal generator for G).

3.3 Preprocessing: The Pre-voting Stage

Each voter V_i, selects n random values $s_{i,j} \in \mathbf{Z}_q$, $j = 1, \ldots, n$, such that $\sum_{j=1}^{n} s_{i,j} = 0$. As we will see, this method will randomize the actual ballots while keeping global consistency; we note that such a technique is typical in various settings, e.g. in re-randomization of individual keys in proactive protocols [OY91]. V_i then, publishes the pairs $\langle R_{i,j}, R'_{i,j} \rangle$, s.t. $R_{i,j} := g^{s_{i,j}}$ and $R'_{i,j} := h_j^{s_{i,j}}$. V_i should prove to any third party that $\log_g R_{i,j} = \log_{h_j} R'_{i,j}$. Using a proof of knowledge introduced in [CP93], this is possible as described in figure 1. Note that this protocol is proven to be zero-knowledge only in the case of a honest verifier (see e.g. [CGS97]), but this is sufficient in our setting.

Prover (V_i)		Verifier
publishes $\langle R_{i,j}, R'_{i,j} \rangle$		
$w \in_R \mathbf{Z}_q$		
$a := g^w, b := h_j^w$	$\xrightarrow{a,b}$	
	$\xleftarrow{c}$	$c \in_R \mathbf{Z}_q$
$r := w + s_{i,j}c$	$\xrightarrow{r}$	$g^r \stackrel{?}{=} a(R_{i,j})^c$
		$h_j^r \stackrel{?}{=} b(R'_{i,j})^c$

Fig. 1. Proving that $\log_g R_{i,j} = \log_{h_j} R'_{i,j}$

The well-known Fiat-Shamir heuristics [FS87] can be used to make the proof non-interactive and ensure that the challenge c is chosen "honestly": if $\mathcal{H}$ is a cryptographically strong hash function (thought of as a random oracle), then c is defined as $\mathcal{H}(I(V_i), R_{i,j}, R'_{i,j}, a, b)$. Consequently V_i publishes $\langle R_{i,j}, R'_{i,j}, a, b, r \rangle$; the verifier computes c using the hash function and checks the two equalities as in the figure. This method ensures that the challenge c is chosen at random (under the assumption that $\mathcal{H}$ is a random oracle hash). When c is chosen using a random oracle hash we will denote the sequence $\langle a, b, r \rangle$ defined as above for the values $R_{i,j}, R'_{i,j}$ and bases g, h_j by PKEQDL$[x : (R_{i,j} = g^x) \wedge (R'_{i,j} = h_j^x)]$. Such proofs will be used in other parts of our voting scheme to ensure that the participants are following the protocol.

Theorem 1. *At any time, after the completion of the pre-voting stage,*
(i) Any third-party can verify that $\log_g R_{i,j} = \log_{h_j} R'_{i,j}$*, for any* i, j*.*
(ii) Any third-party can verify that $\sum_{j=1}^{n} s_{i,j} = 0$*, for any* i*.*
(iii) If at least one voter chose the $s_{i,j}$ *values at random, then the values* $t_j := \sum_{i=1}^{n} s_{i,j}$ *are random elements of* $\mathbf{Z}_q$ *with the property* $\sum_{j=1}^{n} t_j = 0$*.*

Proof. (i) is achieved using the non-interactive version of the proof of knowledge described in figure 1. (ii) can be easily checked by multiplying $R_{i,j}$: it should hold that $\prod_{j=1}^{n} R_{i,j} = 1$ for all $i = 1, \ldots, n$. For item (iii), just note that, $\sum_{j=1}^{n} t_j = \sum_{j=1}^{n} \sum_{i=1}^{n} s_{i,j} = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} = \sum_{i=1}^{n} 0 = 0$. Since each t_j contains a value from each voter V_i, if at least one voter chose the $s_{i,j}$ values at random, this is enough to randomize all t_j values. □

The bulletin board authority for each $j = 1, \ldots, n$ computes the product $R'_j := \prod_{i=1}^{n} R'_{i,j}$ and publishes it on the board. The contents of the bulletin board after the end of the preprocessing are shown in figure 2 (note that in the left hand table, each row when its values are multiplied together the result is the value 1).

$R_{i,j}$:

V_1	:	$g^{s_{1,1}}$	$\ldots$	$g^{s_{1,n}}$	$= 1$
V_2	:	$g^{s_{2,1}}$	$\ldots$	$g^{s_{2,n}}$	$= 1$
$\vdots$		$\vdots$	$\ldots$	$\vdots$	$\vdots$
V_n	:	$g^{s_{n,1}}$	$\ldots$	$g^{s_{n,n}}$	$= 1$
		g^{t_1}	$\ldots$	g^{t_n}	$= 1$

$R'_{i,j}$:

V_1	:	$h_1^{s_{1,1}}$	$\ldots$	$h_n^{s_{1,n}}$
V_2	:	$h_1^{s_{2,1}}$	$\ldots$	$h_n^{s_{2,n}}$
$\vdots$		$\vdots$	$\ldots$	$\vdots$
V_n	:	$h_1^{s_{n,1}}$	$\ldots$	$h_n^{s_{n,n}}$
		$R'_1 = h_1^{t_1}$	$\ldots$	$R'_n = h_n^{t_n}$

Fig. 2. The contents of the Bulletin Board after the preprocessing phase

3.4 Ballot-Casting

Voter V_j reads R'_j from the bulletin board and raises it to α_j^{-1} in order to obtain $h^{s_{1,j}+\ldots+s_{n,j}} = h^{t_j}$. For now, we will assume that votes are either 1 or -1 (yes/no). We note that this is done only for the sake of the exposition as it is possible to modify our protocol in a *direct* and *efficient* manner to support voting between c candidates, or support "0-voting" i.e. abstaining from the formation of the election result (see section 3.7). V_j selects a $v_j \in \{1, -1\}$ at his choice, and publishes the ballot $B_j := h^{t_j} f^{v_j}$.

It is imperative that V_j convinces any third party that he followed the protocol i.e. that the random portion of B_j has been produced by the application of V_j's secret value α_j^{-1} on R'_j (note that the value R'_j is publicly accessible). This can be done as described in figure 3. We note here that one significant difference from previous similar proofs of ballot validity (as those in [CFSY96,CGS97]) is that the value t_j is not known to the voter, and the proof has to rely on the voter (prover) knowing the secret "Diffie-Hellman-key-exchange"-like value α_j.

Prover (V_j)			Verifier
publishes $B_j := h^{t_j} f^{v_j}$			obtains R'_j
$v_j = 1$	$v_j = -1$		
$d_1, r_1, w \in_R \mathbf{Z}_q$	$d_2, r_2, w \in_R \mathbf{Z}_q$		
$a_1 := h^{-d_1} h_j^{r_1}$	$a_1 := h_j^w$		
$a_2 := h_j^w$	$a_2 := h^{-d_2} h_j^{r_2}$		
$b_1 = (B_j f)^{-d_1} (R'_j)^{r_1}$	$b_1 = (R'_j)^w$		
$b_2 = (R'_j)^w$	$b_2 = (B_j / f)^{-d_2} (R'_j)^{r_2}$		
		$\xrightarrow{a_1, a_2, b_1, b_2}$	
		$\xleftarrow{c}$	$c \in_R \mathbf{Z}_q$
$d_2 := c - d_1$	$d_1 := c - d_2$		
$r_2 := w + \alpha_j^{-1} d_2$	$r_1 := w + \alpha_j^{-1} d_1$		
		$\xrightarrow{r_1, r_2, d_1, d_2}$	$c \stackrel{?}{=} d_1 + d_2$
			$h_j^{r_1} \stackrel{?}{=} a_1 h^{d_1}$
			$h_j^{r_2} \stackrel{?}{=} a_2 h^{d_2}$
			$(R'_j)^{r_1} \stackrel{?}{=} b_1 (B_j f)^{d_1}$
			$(R'_j)^{r_2} \stackrel{?}{=} b_2 (B_j / f)^{d_2}$

Fig. 3. Proving that $(\log_{R'_j} B_j f = \log_{h_j} h) \vee (\log_{R'_j} B_j / f = \log_{h_j} h)$

More formally, the protocol of figure 3, is a proof of knowledge in the sense of [CDS94,DDPY94] as it satisfies completeness and special soundness; additionally it is special honest verifier zero knowledge. As before, using the Fiat-Shamir heuristics [FS90] we can make the above protocol non-interactive using random oracle hashing, by choosing $c = \mathcal{H}(I(V_j), B_j, R'_j, a_1, a_2, b_1, b_2)$.

Theorem 2. *The protocol of figure 3 satisfies the following:*
(i) It is a proof of knowledge for the relation

$$(\log_{R'_j} B_j f = \log_{h_j} h) \vee (\log_{R'_j} B_j / f = \log_{h_j} h)$$

(ii) It is Special Honest Verifier Zero Knowledge.
(iii) It ensures that the ballot B_j is formed properly.

Proof. (i) The witness for the relation is the value α_j^{-1} (the secret value of V_j). Completeness can be shown in a straightforward manner.

Regarding special soundness: Given some R'_j, B_j, suppose we have two accepting conversations with the same first move: $(a_1, a_2, b_1, b_2, c, r_1, r_2, d_1, d_2)$ and $(a_1, a_2, b_1, b_2, c', r'_1, r'_2, d'_1, d'_2)$, with $c \neq c'$. For special soundness the goal is to show that a witness can be constructed in polynomial-time. It is easy to check that either $\frac{r_1 - r'_1}{d_1 - d'_1}$ or $\frac{r_2 - r'_2}{d_2 - d'_2}$ is a witness for the relation. At least one of these values is defined since if both $d_1 = d'_1$ and $d_2 = d'_2$ it holds that $c = c'$ (assumed false). Note that a cheating prover can convince the verifier without using a witness by guessing the challenge c sent by the verifier and computing a_1, a_2, b_1, b_2

according to c (instead of prior to receiving c). Such a cheating prover will be detected by the verifier with overwhelming probability $1 - 1/q$.

(ii) The protocol is special honest verifier zero knowledge, because given a random c, if we choose at random r_1, r_2, d_1, d_2 s.t. $c = d_1 + d_2$ the conversation $(h_j^{r_1} h^{-d_1}, h_j^{r_2} h^{-d_2},\ (R'_j)^{r_1}(B_j f)^{-d_1}, (R'_j)^{r_2}(B_j/f)^{-d_2}, c, r_1, r_2, d_1, d_2)$ is an accepting conversation that it is indistinguishable from the accepting conversations generated by the honest prover and the honest verifier.

(iii) From the relation $(\log_{R'_j} B_j f = \log_{h_j} h) \vee (\log_{R'_j} B_j/f = \log_{h_j} h)$, if we denote by x the value $\log_{h_j} h$ then it follows that either $B_j = (R'_j)^x f$ or $B_j = (R'_j)^x / f$ is true which is exactly how V_j is supposed to form the ballot B_j. □

3.5 Administrating and Terminating the Election

The bulletin board authority is responsible for the administration of the election. When the pre-voting stage is completed the bulletin board authority signs the bulletin board and announces the start of the election process. It is imperative that the bulletin board authority prevents reading of the cast ballots as the election progresses in order to ensure fairness. This is because the last voter to cast a ballot is able to compute the election result before choosing his/her vote and casting the final ballot. Fairness can be ensured as follows: the bulletin board authority participates in the pre-voting stage acting as one of the voters, say voter i_0. The bulletin board authority does not cast a ballot; instead when the deadline of the election process is reached it publishes $B_{i_0} := (R'_{i_0})^{\alpha_{i_0}^{-1}} = h^{t_{i_0}}$ together with the non-interactive proof of knowledge PKEQDL$[x : (h = h_{i_0}^x) \wedge (B_{i_0} = (R'_{i_0})^x)]$ (see section 3.3) that ensures that the correct value is published — note that this amounts to a publicly verifiable 0-vote. The bulletin board authority also signs the contents of the bulletin board, thus officially terminating the election process and prohibiting the deposition of any other ballots.

Given the way the voters' ballots are formed and theorem 1(iii), it is easy to see that:

Proposition 3. *The administration of the election by the Bulletin Board Authority as described above ensures fairness.*

3.6 The Final Stage: Self-tallying and Corrective Fault-Tolerance

All voters participate: Self-tallying
If all voters participate in the election, then computing the final tally can be done by any (casual) third party in optimal time: $n - 1$ multiplications in G and an exhaustive search of $2n$ steps, worst-case. Since $\sum_{j=1}^{n} t_j = 0$ (see theorem 1,iii) it holds that the tally $T := \prod_{j=1}^{n} B_j = f^{\sum_{j=1}^{n} v_j}$ (note that v_{i_0} the "vote" cast by the bulletin board authority is equal to 0). Because $T \in \{f^{-n+1}, \ldots, f^{n-1}\}$ it can be inverted by a brute-force algorithm that will check all the possible values.

By a baby-step giant-step method the inversion can be made in $\mathcal{O}(\sqrt{n})$ group operations.

Corrective fault tolerance

In order to achieve corrective fault-tolerance, when some voters stop participating at some phase of the protocol, the remaining active voters must react to reveal the shares that were intended for them. Here we deal with two cases: (i) when some registered voters do not participate in the pre-voting stage, and (ii) when some voters do not cast a ballot before the deadline of the election. We note here that corrective fault-tolerance is not intended to capture all fault possibilities of the active participants, and there can be catastrophic cases for which the protocol must restart (with the faulty participants deactivated) for example when some of the voters that cast a ballot do not participate in the corrective phase. Note though, that whenever the majority is honest there is a way to prevent a restart of the protocol by Byzantine agreement procedures — an issue which is beyond the scope of this paper. Nevertheless, our protocol is intended for the small scale boardroom model, where malicious failures (where members of the board misbehave publicly) are not typical, and also in this case, a restart of the protocol is not prohibitive.

Nevertheless, as we will see in section 4.1, corrective fault tolerance does provide significant increased of robustness, when compared to all previous election schemes in the Perfect Ballot Secrecy setting. Next we outline the two procedures which constitute corrective fault tolerance.

(i) Some registered voters do not participate in preprocessing. The method presented here is also useful when some voters that participated in the preprocessing phase prefer not to vote. This is important in the batched version of the self-tallying scheme (see section 4.2). Denote the set of voters whose shares should be cancelled by S, and the set of remaining voters by $\overline{S}$.

The corrective phase is as follows: each voter V_k, $k \in \overline{S}$, publishes $R''_k := h_k^{\sum_{j\in S} s_{k,j}}$, together with the non-interactive proof of knowledge PKEQDL$[x : (h_k^x = R''_k) \wedge (g^x = \prod_{j\in S} R_{k,j})]$ (see section 3.3). The bulletin board authority modifies the R'_k values (that were computed at the end of the preprocessing phase) as follows:

$$\forall k \in \overline{S} \quad R'_k := R'_k R''_k / \prod_{i\in S} R'_{i,k}$$

Then, the protocol proceeds normally with ballot-casting (section 3.4). It is easy to see that the values $t_k := \log_{h_k} R'_k$ for $k \in \overline{S}$ satisfy the properties of the t-values in theorem 1(iii): $\sum_{k\in\overline{S}} t_k = 0$ and $\{t_k\}_{k\in\overline{S}}$ are random elements of $\mathbf{Z}_q$ provided that at least one voter V_k, $k \in \overline{S}$ chose the shares $s_{k,j}$ at random.

(ii) Some voters do not cast a ballot. Denote the set of voters that did not cast a ballot by S', and the set of remaining voters by $\overline{S'}$. Each voter V_k, $k \in \overline{S'}$, publishes $e_k := \sum_{j\in S'} s_{k,j}$ and $\Phi_k := (\prod_{j\in S'} R'_{j,k})^{\alpha_k^{-1}}$.

The value e_k can be universally verified by checking that $g^{e_k} = \prod_{j \in S'} R_{k,j}$ for all $k \in \overline{S'}$. The correctness of the value Φ_k can be universally verified by having the voter V_k publish the non-interactive proof of knowledge PKEQDL$[x : (h = h_k^x) \wedge (\Phi_k = (\prod_{j \in S'} R'_{j,k})^x)]$ (see section 3.3). After the completion of the corrective round the tally computation can be performed by any third party as follows: $T := \prod_{k \in \overline{S'}} B_k h^{e_k} (\Phi_k)^{-1}$. It is easy to see that $T \in \{f^{|\overline{S'}|}, \ldots, f^{-|\overline{S'}|}\}$; inverting T is done in a brute-force manner as before.

We note here that these two corrective phases, are not exclusive and both can be implemented in a single execution of a protocol.

3.7 Multi-way Elections

There are many standard ways to extend our voting scheme from yes/no voting to 1-out-of-c voting for c candidates (for a constant c) in an efficient manner[1]. We describe one such possibility following ideas from Cramer, Gennaro and Schoenmakers [CGS97]. The resulting scheme has the same complexity in all respects, with the exception that the proof of validity of the ballot is increased by the factor c, and the exhaustive search at the tallying phase requires searching in the augmented possible tally value space.

Instead of f, in the initialization stage the values $f_1, \ldots, f_c \in G$ are given for use by all participants (where the $\log_h f_\ell$ is not known to any party). V_j casts a vote $v_j \in \{1, \ldots, c\}$ by publishing the ballot $h^{t_j} f_{v_j}$. The proof of ballot validity is modified as shown in figure 4.

Theorem 3. *The protocol of figure 4 satisfies the following:*
(i) It is a proof of knowledge for the relation $\vee_{\ell=1}^{c} (\log_{R'_j} B_j / f_\ell = \log_{h_j} h)$.
(ii) It is Special Honest Verifier Zero Knowledge.
(iii) It ensures that the ballot B_j *is formed properly.*

The election protocol is otherwise exactly the same to the yes/no case. In the final stage the product $T_1 \ldots T_c$ is revealed where $T_\ell \in \{f_\ell^0, \ldots, f_\ell^{n-1}\}$. Revealing the tallies, requires a total of n^{c-1} search steps in the worst case. By using a baby-step giant-step method it is possible to reduce the time needed to $\mathcal{O}(\sqrt{n}^{c-1})$ group operations (see also [CGS97]) . We would like to further note that the searching task can be arbitrarily partitioned among many processors in a distributed environment.

4 The Voting Scheme

Stage 1. All participants run the initialization as described in section 3.2. Voters register with the bulletin board authority that checks their eligibility, and gain access to the bulletin board.

[1] To go beyond constant c one needs either to use another method than homomorphic-based encryption, or base the scheme on sharing Paillier's cryptosystem [Pai99] as in [BFPPS01] which we can employ as well, at the expense of losing some of our scheme's properties.

Prover (V_j)		Verifier
publishes $B_j := h^{t_j} f_{v_j}$		obtains R'_j
for $\ell \in \{1, \ldots, c\} - \{v_j\}$		
$d_\ell, r_\ell \in_R \mathbf{Z}_q, r_{v_j}, w \in_R \mathbf{Z}_q$		
$a_\ell = h^{-d_\ell} h_j^{r_\ell}$		
$a_{v_j} = h_j^w$		
$b_\ell = (B_j/f_\ell)^{-d_\ell} (R'_j)^{r_\ell}$		
$b_{v_j} = (R'_j)^w$		
	$\{a_\ell\}_\ell, \{b_\ell\}_\ell \longrightarrow$	
	$\longleftarrow c$	$c \in_R \mathbf{Z}_q$
$d_{v_j} := c - (\sum_{\ell \neq v_j} d_\ell)$		
$r_{v_j} := w + \alpha_j^{-1} d_{v_j}$		
	$\{r_\ell\}_\ell, \{d_\ell\}_\ell \longrightarrow$	$c \stackrel{?}{=} \sum_\ell d_\ell$
		for $\ell = 1, \ldots, c$,
		$h_j^{r_\ell} \stackrel{?}{=} a_\ell h^{d_\ell}$
		$(R'_j)^{r_\ell} \stackrel{?}{=} b_\ell (B_j/f_\ell)^{d_\ell}$

Fig. 4. Multi-way Elections: Proving that $\vee_{\ell=1}^{c} (\log_{R'_j} B_j/f_\ell = \log_{h_j} h)$

Stage 2. The voters together with the bulletin board authority execute the preprocessing stage of section 3.3. At the end of the preprocessing stage, if some registered voters failed to participate the corrective step of section 3.6(i) is employed. Subsequently the bulletin board authority signs the bulletin board and officially starts the election.

Stage 3. Voters publish their ballots as described in section 3.4 in the designated space for them in the bulletin board. Note that this process is not anonymous and as a result double-voting is eliminated. When the deadline for the election is reached the bulletin board casts the "final vote" and thus opens the election results (see section 3.5).

Stage 4. If all voters who received shares in the preprocessing phase cast a ballot, the election tally is recoverable by any third party as described in section 3.6 (self-tallying). If not all voters participated the corrective fault tolerance round of section 3.6(ii) is employed.

Complexity. Stage 1 requires $\mathcal{O}(1)$ computation and communication by all participants. In stage 2, each voter spends $\mathcal{O}(n)$ for computing and communicating the preprocessing shares. In stage 3, the voter computes the ballot and the proof of validity that needs $\mathcal{O}(1)$ computation and $\mathcal{O}(1)$ communication. Stage 4, if all voters that received shares at stage 2 cast a ballot, requires $n-1$ multiplications and a brute-force step that takes $2n$ steps worst-case, which can be reduced to $\mathcal{O}(\sqrt{n})$. The corrective rounds of sections 3.6 have the following complexity: if

b voters stop participating, each active voter needs to perform $b-1$ operations and to publish the resulting corrective value that requires $\mathcal{O}(1)$ communication.

Assuming the existence of an homomorphic encryption with an associated discrete log problem which is secure and a random oracle hash, it holds that:

Theorem 4. *The described protocol is a voting-scheme that satisfies privacy, fairness, universal-verifiability, corrective fault-tolerance, dispute-freeness, self-tallying and perfect ballot secrecy.*

4.1 Comments on PBS Voting and Dispute-Freeness

In a voting-scheme with perfect-ballot secrecy, the actual vote of a participant can only be revealed if all the remaining voters collude against him. In our case Perfect Ballot Secrecy is ensured computationally. Our voting scheme can be modified so that Perfect Ballot Secrecy is information theoretic however the resulting scheme will not be dispute-free anymore.

Other voting schemes based on homomorphic encryption can be modified so that they achieve Perfect Ballot Secrecy. Usually the computational/ communication cost is substantial. The general idea is to make the set of authorities coincides with the set of voters and to set the threshold to n (for schemes that are based on secret sharing). This choice disables the robustness property as seen in proposition 2. Note that mix-net based schemes are not necessarily excluded from the PBS setting (since it may be possible for the voters to simulate the mix-net), but the homomorphic encryption setting seems to be more natural as a base for PBS. In any case, a mix-net PBS scheme would face similar robustness problems since *all* voters would need to be active in the mixing phase in order to ensure perfect ballot secrecy.

Let us discuss the notion of dispute-freeness and its importance in the PBS homomorphic-encryption setting. The schemes of [CFSY96] and [CGS97] include phases that are not publicly verifiable and therefore they are not dispute-free: the first uses private channels between voters and authorities whereas the second includes a key-generation phase between authorities that is not publicly-verifiable (note that we do not claim that lack of dispute-freeness is necessarily inherent in these schemes). Since in PBS voting the voters play the role of the authorities this can lead to voters accusing other voters of invalidity of actions, which is certainly not desirable. For this reason we will focus our discussion of PBS schemes (and in general of voting without authorities) on schemes that are dispute-free i.e. [Sch99] and our voting scheme.

The [Sch99]-scheme can be transformed to the PBS setting: choosing the threshold t to be n and letting the voters play the role of the talliers. The complexity of the voter then becomes $\mathcal{O}(n^2)$ before ballot casting and all voters have to be active in the post ballot casting stage spending $\mathcal{O}(n)$ computation. Due to the fact that *all* voters need to be active in the tallying phase, the scheme is not robust (as it is the case for all schemes that base fault-tolerance on secret-sharing, in the PBS setting, see proposition 2). Note though that it is possible to achieve some form of corrective fault-tolerance by adding additional

rounds of communication and computation, but this amounts to implementing our own corrective steps (which become more complex when put on top of the [Sch99]-scheme rather than as used in our scheme).

The complexity of our voting scheme is substantially better than the PBS-version of the [Sch99] scheme. In our case in the pre-voting stage, each voter needs computation and communication $\mathcal{O}(n)$. Tallying is optimal as it takes $3n$ steps, without the participation of the voters. Corrective fault tolerance costs $\mathcal{O}(b)$ group operations for a participant, where b is the number of faults.

As we pointed out before, most homomorphic encryption based election schemes can be readily modified into the PBS setting. With no exception this will force all voters to be active in the post-ballot-casting phase and in fact each voter will have to perform an $\mathcal{O}(n)$ computation at this stage. If one of the voters fails to participate in the post-ballot-casting phase the entire election fails. This is much too late in the process to allow such failures. In contrast, our election scheme optimizes this setting: if all voters participate in the election there is no need for them to participate in the post-ballot-casting phase at all. Moreover if some voters do not participate in the voting phase the election does not necessarily fail since we have shown a set of corrective measures that can be executed. This delicate fault tolerance distinction is an important achievement of our scheme. Performance-wise, what our active corrective fault tolerance requires from active voters is not $\mathcal{O}(n)$ but $\mathcal{O}(b)$ work for b absent voters, namely work proportional to the absentees and not to the whole voter population (what the distributed computing literature calls "adaptive fault tolerance").

4.2 Batched Boardroom PBS-Elections

Batching a series of elections is one of the ways to maximize the gain from our voting scheme.

Phase 1. All board members execute the pre-voting stage for several elections in the beginning of a time-period. The work of each participant for each of the elections is linear in the total members of the board. The output of this stage will be kept for later use.

Phase 2. If all members of the board are going to vote in a particular election, the election is held directly as in sections 3.4 and 3.6. If not all members of the board are "present" in the election (something that is known in advance) the values R'_j are properly modified so that the self-tallying property is preserved as described in section 3.6(i). If some voters fail to cast a ballot the corrective phase of section 3.6(ii) is executed.

The complexity of the batched protocol is as follows: The pre-voting stage can be executed off-line ahead of time preparing for coming L elections. It requires $\mathcal{O}(Ln)$ computation for each board member and $\mathcal{O}(Ln^2)$ storage space in total. Each election itself requires $\mathcal{O}(1)$ per participant and then any passive third party can compute the final tally with $n-1$ multiplications and a worst-case

of $2n$ steps in the brute-force stage. The corrective phase requires linear (in the number of absentees) computations from any active participant. Thus, we get:

Theorem 5. *The described batched election protocol is a voting-scheme that satisfies privacy, fairness, universal-verifiability, corrective fault-tolerance, dispute-freeness, self-tallying, perfect ballot secrecy where the on-line cost is that of ballot-casting: constant in the number of participants.*

4.3 Adapting to Large Scale: The Non-PBS Version

If Perfect Ballot Secrecy is not an objective our scheme can be readily modified to an authority-based self-tallying election scheme that is more suitable for the large scale setting. In the modified scheme the voter needs to be active only in the registration phase and the ballot-casting phase.

Stage 1. As in the PBS-scheme, where the set of participants includes a number of authorities $A_1, \ldots, A_m$.

Stage 2. Only the authorities are active in the preprocessing phase, with each A_i issuing the values $s_{i,1}, \ldots, s_{i,n}$ as described in section 3.3.

Stage 3. Identical to the ballot-casting stage as described in the PBS-version.

Stage 4. If all voters who received shares in the preprocessing phase cast a ballot, the election tally is recoverable by any third party as described in section 3.6 (self-tallying). If not all voters participate, corrective fault tolerance has the authorities publishing the shares of the voters that did not cast a ballot.

The computation/communication for each authority is linear in n, whereas the ballot-casting stage remains of constant time for each voter. It is easy to see that:

Theorem 6. *The described protocol is a voting-scheme that satisfies privacy (assuming not all authorities collude), fairness, universal-verifiability, corrective fault-tolerance, dispute-freeness, and self-tallying.*

References

Abe99. Masayuki Abe, *Mix-Networks on Permutation Networks*, ASIACRYPT 1999.

BFPPS01. Olivier Baudron, Pierre-Alain Fouque, David Pointcheval, Guillaume Poupard and Jacques Stern, *Practical Multi-Candidate Election system*, In the Proceedings of PODC 2001.

Ben87. Josh Benaloh, *Verifiable Secret-Ballot Elections*, PhD Thesis, Yale University, 1987.

BY86. Josh Benaloh and Moti Yung, *Distributing the Power of a Government to Enhance the Privacy of Voters*, PODC 1986.

BT94. Josh Benaloh and Dwight Tuinstra, *Receipt-Free Secret-Ballot Elections*, STOC 1994.

B89. Colin Boyd, *A New Multiple Key Cipher and an Improved Voting Scheme*, EUROCRYPT 1989.

Bra99. Stefan Brands, *Rethinking Privacy*, Ph.D. thesis, pages 230-231.

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

Cha88. David Chaum, *Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA* EUROCRYPT 1988.

CP93. David Chaum and Torben P. Pedersen, *Wallet Databases with Observers*, CRYPTO 1992.

CF85. Josh D. Cohen (Benaloh) and Michael J. Fischer, *A Robust and Verifiable Cryptographically Secure Election Scheme*, FOCS 1985.

CGS97. Ronald Cramer, Rosario Gennaro and Berry Schoenmakers, *A Secure and Optimally Efficient Multi-Authority Election Scheme*, EUROCRYPT 1997.

CDS94. Ronald Cramer, Ivan Damgård and Berry Schoenmakers, *Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols*, CRYPTO 1994.

CFSY96. Ronald Cramer, Matthew K. Franklin, Berry Schoenmakers and Moti Yung, *Multi-Autority Secret-Ballot Elections with Linear Work*, EUROCRYPT 1996.

DJ00. Ivan Damgård and Mats Jurik, *A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System*, Public Key Cryptography 2001, pp. 119-136.

DLM82. Richard A. DeMillo, Nancy A. Lynch, Michael Merritt, *Cryptographic Protocols*, STOC 1982: pp. 383-400.

DDPY94. Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, Moti Yung, *On Monotone Formula Closure of SZK*, FOCS 1994.

FS90. Uriel Feige and Adi Shamir, *Witness Indistinguishable and Witness Hiding Protocols*, STOC 1990.

FS87. Amos Fiat and Adi Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*, CRYPTO 1986.

FPS00. Pierre-Alain Fouque, Guillaume Poupard and Jacques Stern, *Sharing Decryption in the Context of Voting or Lotteries*, In the Proceedings of Financial Cryptography 2000.

FOO92. Atsushi Fujioka, Tatsuaki Okamoto and Kazuo Ohta: *A Practical Secret Voting Scheme for Large Scale Elections*, ASIACRYPT 1992.

GJKR99. Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk and Tal Rabin, *Secure Distributed Key Generation for Discrete-Log Based Cryptosystems* EUROCRYPT 1999.

HS00. Martin Hirt and Kazue Sako, *Efficient Receipt-Free Voting Based on Homomorphic Encryption*, EUROCRYPT 2000.

Jak99. Markus Jakobsson, *Flash Mixing*, Principles of Distributed Computing (PODC), 1999.

KMO01. Jonathan Katz, Steven Myers, and Rafail Ostrovsky, *Cryptographic Counters and Applications to Electronic Voting*, EUROCRYPT 2001.

Mer83. Michael Merrit, *Cryptographic Protocols*, Ph.D. Thesis, Georgia Institute of Technology 1983.

NSS91. Hannu Nurmi, Arto Salomaa, and Lila Santean, *Secret Ballot Elections in Computer Networks.*, Computers & Security 36, 10 (1991), 553-560.

OKST97. Wakaha Ogata, Kaoru Kurosawa, Kazue Sako and Kazunori Takatani, *Fault tolerant anonymous channel*, In the Proceedings of ICICS '97, LNCS No. 1334, pp. 440–444, 1997.

Oka97. Tatsuaki Okamoto, *Receipt-Free Electronic Voting Schemes for Large Scale Elections*, Workshop on Security Protocols, 1997.

OY91. R. Ostrovsky and M. Yung, *How to withstand mobile virus attacks*, ACM Symposium on Principles of Distributed Computing (PODC), 1991, pp. 51-61.

Pai99. Pascal Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, EUROCRYPT 1999.

PIK94. Choonsik Park, Kazutomo Itoh and Kaoru Kurosawa, *Efficient Anonymous Channel and All/Nothing Election Scheme*, EUROCRYPT 1993.

PW92. Birgit Pfitzmann and Michael Waidner, *Unconditionally Untraceable and Fault-tolerant Broadcast and Secret Ballot Election*, Hildesheimer Informatik-Berichte, Institut für Informatik, Universität Hildesheim, 1992.

Sak94. Kazue Sako, *Electronic Voting Schemes, Allowing Open Objection to the Tally*, In the Transactions of the Institue of Electronics, Information, and Communication Engineers, volume E77-A, n. 1, pp. 24-30, 1994.

SK94. Kazue Sako and Joe Kilian, *Secure Voting Using Partially Compatible Homomorphisms,* CRYPTO 1994.

SK95. Kazue Sako and Joe Kilian, *Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth*, EUROCRYPT 1995.

Sch99. Berry Schoenmakers, *A Simple Publicly Verifiable Secret Sharing Scheme and its Applications to Electronic Voting*, CRYPTO 1999.

Yao82. Andrew C. Yao, *Protocols for Secure Computations*, Proc. 23 rd IEEE Symp. on Foundations of Computer Science, Chicago, IL (Nov. 1982), 160–164. 17.