# Electronic Jury Voting Protocols

Alejandro Hevia* Marcos Kiwi†

**Abstract**

This work elicits the fact that all current proposals for electronic voting schemes disclose the final tally of the votes. In certain situations, like jury voting, this may be undesirable. We present a robust and universally verifiable Membership Testing Scheme (MTS) that allows, among other things, a collection of voters to cast votes and determine whether their tally belongs to some pre–specified set (e.g., exceeds a given threshold) — our scheme discloses no additional information than that implied from the knowledge of such membership. We discuss several extensions of our basic MTS. All the constructions presented combine features of two parallel lines of research concerning electronic voting schemes, those based on MIX–networks and in homomorphic encryption.

**Keywords** — Membership testing, MIX–networks, majority voting, election scheme.

## 1 Introduction

In a typical trial by jury in the United States, twelve jurors deliberate in private. A foreman appointed by the judge among the jurors presides the deliberations. Jurors might be called upon to decide on several different counts according to a policy which may be complicated. Nevertheless, the simplest and most important jury verdicts are of the binary type, e.g., innocent/guilty. In criminal cases unanimity is required in order to reach a verdict. In civil cases there are different standards, nine out of twelve votes are representative numbers. Jury deliberations proceed in discussion rounds followed by voting rounds. Voting is performed by raising hands. Hence, a typical requirement of an election protocol, privacy of the votes, is not achieved. This opens the possibility of biases on decisions due to jurors fear of rejection, a posteriori reprisals by interested parties, and/or follow the leader kind of behavior. In fact, just knowledge of tallies can cause undesirable consequences like follow the pack kind of behavior among jurors.

A ballot box system could be implemented in order to guarantee privacy. A subset of the jury might be held responsible for tallying the votes and communicating to the others whether a verdict has been reached. Still, this discloses the final tally to a subset of the jury and allows them to manipulate the deliberation process. An outside third party (e.g., a judge, government employee, etc.) could be responsible for tallying the votes, but this would cast doubts on the whole process since it allows for outside jury manipulation, could cause undesirable leaks on how the jury is leaning, etc.

*Dept. of Computer Science & Engineering, U. of California, San Diego, CA, and Dept. Cs. de la Computación, U. Chile, `ahevia@cs.ucsd.edu`. Partially supported by Conicyt via Fondap in Applied Mathematics 1999–2000 and Fondecyt No. 1981182.

†Dept. Ing. Matemática, Ctr. de Modelamiento Matemático, UMR 2071 U. Chile–CNRS, `mkiwi@dim.uchile.cl`. Gratefully acknowledges the support of Conicyt via Fondecyt No. 1981182 and Fondap in Applied Mathematics 1999–2000.

We provide an electronic drop in procedure for jury voting in the presence of a curious media, interested parties, dishonest court employees, and conflictive jury members, that reveals nothing besides whether the final tally exceeds or not a given threshold value.

We stress that we do not question the adequacy of the way in which juries deliberate. There are good reasons to encourage jurors to express clearly and openly their opinions. The point is that the way in which juries deliberate is just one familiar example, among many, where it is clear that the voting procedure itself has an effect on the final outcome. In particular, our work is motivated by the observation that voting procedures that disclose final tallies may be undesirable. This situation occurs when small groups wish to reach by majority vote a yes/no type decision on a particular issue: e.g., whether to accept or reject a paper submitted to a cryptology conference — the cryptographers program committee problem, to confirm or not someone as president of a committee or chair of a department, a program committee deciding whether or not to send an invitation to a speaker, business partners trying to decide whether to go forth with a given investment.

The first electronic voting scheme proposals focused on trying to break the correspondence between the voters and the vote casted. Afterwards, several other desirable properties of electronic voting schemes (besides correctness, privacy, and efficiency) were identified, e.g., robustness, availability, non–duplication, universal verifiability, non–coercibility. Electronic voting protocols satisfying different subsets of the latter properties were designed. Nevertheless, all of them reveal the final vote tally. In this work we propose a cryptographic procedure for addressing this problem and stress its relevance by describing a different application.

Before presenting this work's contributions we begin by discussing past work.

### 1.1 Related work

Voting schemes where one wants only one bit of information regarding the outcome, like the ones discussed in the previous section, can be cast in the framework of secure multi–party computation. Thus, plausibility results, asserting that such voting schemes can be in principle constructed, can be obtained. Indeed, the application of general secure multi–party computation techniques like the ones proposed in [GMW86, CCD87, BGW88] yield such constructions. Unfortunately, they yield constructions that do not exhibit some of the properties one desires of an electronic voting scheme (e.g., non–interaction among voters).

Electronic voting schemes are one of the prime examples of secure multi–party computation. Thus, it is not surprising that they have been intensively studied. The first electronic election scheme in the literature was proposed by Chaum [Cha81]. His work is based on a realization of a computational secure anonymous channel called the MIX–network. Anonymous channels and election schemes are closely related. Indeed, an anonymous channel hides the correspondence between senders and receivers. An election scheme requires hiding the correspondence between the voters and their votes. Since Chaum's [Cha81] work, several electronic election scheme proposals based on untraceability networks have been described in the literature, among the earlier ones see [Cha88, Boy89, PIK93, SK95]. More recent proposals of these type are those of Abe [Abe98], Jakobsson [Jak98, Jak99], and Jakobsson and Juels [JJ99]. (For actual implementations of MIX–networks see [SGR97] and the references therein.)

In contrast to the schemes mentioned in the previous paragraph, the ones introduced in [CF85, BY86, Ben87] do not rely on the use of anonymous channels. In these schemes ballots are distributed over a number of tallying authorities through a special type of broadcast channel. Rather than hiding the correspondence between the voter and his ballot, the value of the vote is hidden. Among these latter type of schemes are [Ive91, SK94, BT94]. More recent proposals are those of Cramer et al. [CFSY96, CGS97] and Schoenmakers [Sch99]

For other related work on electronic voting schemes see [FOO92, OO89, NR94, HS00]

### 1.2 Our Contributions

This work's first contribution is that it elicits the fact that all current proposals for electronic voting schemes disclose the final tally of the votes. As discussed above this may be undesirable in some situations. Our main technical contribution is a cryptographic protocol to which we refer as *Membership Testing Scheme (MTS)*. Given a fixed sequence of integers $c_1, \ldots, c_n$ and sets $S_1, \ldots, S_n$, it allows a collection of parties $P_1, \ldots, P_n$ to cast values $v_1, \ldots, v_n$, where $v_i \in S_i$, and determine whether $\sum_i c_i v_i$ belongs to some pre–specified set $S$.

Based on our MTS we obtain a drop in replacement electronic procedure for a civil case jury voting protocol by letting $n = 12$, $c_1 = \ldots = c_n = 1$, $S_1 = \ldots = S_n = \{0, 1\}$, and $S = \{9, 10, 11, 12\}$ (simpler schemes can be devised for criminal type trials, so we will focus on the more challenging civil case type trials). For the sake of simplicity of exposition, throughout we informally discuss our results in the terminology of jury systems. Thus, for notational and mnemonic purposes we refer to parties $P_1, \ldots, P_n$ as voters and denote them by $V_1, \ldots, V_n$, to the values $v_1, \ldots, v_n$, as votes, and to $\sum_i c_i v_i$ as the tally.

The schemes we propose satisfy some subset of the following properties:

- Eligibility: Only authorized voters can vote. No one can vote more than once.
- Correctness: If all participants are honest, the correct output will be generated.
- Robustness: The system can recover from the faulty or malicious behavior of any (reasonably sized) coalition of participants.
- Computational Privacy: A voter ballot's content will be kept secret from any (reasonably sized) coalition of parties that does not include the voter.
- Universal Verifiability: Ensures that any party, including a passive observer, can check that ballots are correctly cast, only invalid ballots are discarded, and the published final tally is consistent with the correctly cast ballots.
- No–duplication: No one can duplicate anyone else's vote.
- Receipt–Freeness: No voter can prove to a third party that he/she has cast a particular vote.

In our scheme the voters send in a ballot identical to those proposed in [CGS97], i.e., an ElGamal ciphertext representing his/her vote plus a proof that the ciphertext is indeed a valid ballot. Hence, as in [CGS97], both the computational and communicational complexity of the voter's protocol is linear in the security parameter $k$ — thus optimal.[1] Moreover, for any reasonable security parameter, the voters' protocol remains the same even if the number of voters varies. The work performed by each authority is $O((m|S| + n)k)$. Moreover, the computational complexity of verifying each authority's work is proportional to the work performed by each authority. As in [CGS97] the work needed to verify that a voter sent in a well formed ballot is $O(k)$, per voter.

[1] Throughtout, a modular multiplication of $O(k)$ bit sized numbers will be our unit with respect to which we measure computational costs.

Our MTS proposal combines features of two parallel lines of research concerning electronic voting schemes, those based on MIX–networks (a la [Cha81]) and in homomorphic encryption schemes (a la [CF85, BY86, Ben87]). We use homomorphic (ElGamal) encryption in order to hide the vote tallies. We rely on special properties of the ElGamal cryptosystem in order to perform an equality test between the tally and members of $S$. We use MIX–networks (ElGamal based) in order to hide the value of the member of $S$ involved in each equality test. To the best of our knowledge, the only other cryptographic scheme which relies on homomorphic encryption schemes and MIX–networks is the recently proposed scheme of Hirt and Sako [HS00]. But, our MTS combines both theses schemes in a novel way. Indeed, Hirt and Sako's proposal uses a MIX–network in order randomly permute, for each voter, potential ballots. Our MTS relies on MIX–networks in order to randomly permute the elements of the pre–specified set $S$ on which one desires to test membership.

The applications we provide for our MTS constitute novel uses of MIX–networks. A feature of these applications is that they rely on the capacity, that the overwhelming majority of MIX–network proposals exhibit, to randomly permute and encrypt a list of ElGamal ciphertexts. On the contrary, they do not use the decryption capabilities that accompany most MIX–network proposals.

We propose several implementations of a MTS. Our first proposal relies on the homomorphic encryption based electronic election scheme of Cramer, Gennaro and Schoenmakers [CGS97] and the MIX–network of Abe [Abe98]. We also discuss alternative implementations of our MTS based on the work of Jakobsson and Desmedt and Kurosawa [Jak98, DK00] as opposed to that of Abe [Abe98]. Our different MTS implementations exhibit different properties depending on the previous work we use to build them.

### 1.3 Organization

In Sect. 2, we discuss the building blocks on which our basic MTS proposal relies. In Sect. 3, we describe and analyze our MTS and use it for building an electronic drop in replacement for a jury voting protocol that reveals nothing besides the fact that the final tally exceeds or not a given threshold. In Sect. 5 and Sect. 6 we discuss variants and other applications of our basic scheme. We conclude in Sect. 7 discussing a feature of all of the MTSs that we propose and some desirable future developments.

## 2 Preliminaries

We work in the same model introduced by Benaloh et al. [CF85, BY86, Ben87, CGS97], where participants are divided into $n$ voters $V_1, \ldots, V_n$ and $m$ authorities $A_1, \ldots, A_m$ called active parties. Al parties are limited to have polynomial–bounded computational resources and have access to a so called bulletin board whose characteristics we describe below.

In the sequel we assume that a designated subset of active participants on input $1^k$ where $k$ is a security parameter, jointly generate the following system parameters: large $k$ bit long primes $p$ and $q$ such that $q$ divides $p-1$, and generators $g$ and $h$ of a multiplicative subgroup $G_q$ of order $q$ of $\mathbb{Z}_p^*$. One way of collectively generating these system parameters is letting participants run a copy of the same probabilistic algorithm, where the algorithm's coinflips are generated mutually at random.

Conventions: Henceforth, unless otherwise specified, all arithmetic is performed modulo $p$ except for arithmetic involving exponents which is performed modulo $q$. Throughout this paper, $x \in_R \Omega$ means that $x$ is chosen uniformly at random from $\Omega$. Furthermore, negligible and overwhelming probability correspond to probabilities that are at most $\nu(k)$ and $1-\nu(k)$ respectively, where $\nu(k)$

is a function vanishing faster than the inverse of any polynomial in the security parameter $k$. A non–negligible probability is said to be significant.

### 2.1 Building Blocks

BULLETIN BOARD: The communication model used in our MTS consists of a public broadcast channel with memory, usually referred too in the literature as bulletin board. All messages that pass through this communication channel can be observed by any party including passive observers. Nobody can erase, alter, nor destroy any information. Every active participant can post messages in his own designated section of a bulletin board. This requires the use of digital signatures to control access to distinct sections of the bulletin board. Here we assume a public–key infrastructure is already in place. This suffices for computational security.

Note that it is implicitly assumed that denial–of–service attacks are excluded from consideration (see [CGS97] for a discussion of how to implement a bulletin board in order to achieve this).

DISTRIBUTED KEY GENERATION PROTOCOL (DKG): A DKG protocol allows parties $A_1, \ldots, A_m$ to respectively generate private outputs $s_1, \ldots, s_m$, called shares, and a public output $y = g^s$ such that the following requirements are met:

- **Correctness:** There is an efficient procedure that on input at least $t+1$ shares submitted by honest parties and the public information produced by the DKG protocol, outputs the unique secret value $s$, even if up to $t$ shares are submitted by faulty parties. All honest parties coincide on the public key $y = g^s$ and $s \in_R \mathbb{Z}_q$.

- **Secrecy:** No information on $s$ can be learned by the adversary except what is implied by the value $y = g^s$ (for a more formal definition in terms of simulatability see [GJKR99]).

The first DKG protocol was proposed by Pedersen [Ped91]. Henceforth in this work, DKG refers to the protocol presented in [GJKR99] and shown to be secure in the presence of an active adversary that can corrupt up to $t < n/2$ parties.

ELGAMAL ENCRYPTION AND ROBUST (THRESHOLD) PROOF OF EQUALITY OF ENCRYPTIONS: Our MTS relies on a robust threshold version of the ElGamal cryptosystem [ElG85] proposed in [CGS97]. Recall that in ElGamal's cryptosystem the sender chooses $r \in_R \mathbb{Z}_q$, and encrypts the message $m \in G_q$ as $(\alpha, \beta) = (g^r, y^r m)$ where $y = g^s$ is the public key and $s$ is the secret key. In a robust threshold version of the ElGamal cryptosystem, the secret key and public key are jointly generated by the intended ciphertext recipients by means of a DKG protocol like the one described above.

A robust threshold ElGamal cryptosystem has a feature on which all our MTS proposals rely. This property allows checking whether a ciphertext encodes the plaintext 1 without either decrypting the message nor reconstructing the secret $s$. Indeed, assume $(\alpha, \beta)$ is an ElGamal encryption of message $m$, that is $(\alpha, \beta) = (g^r, y^r m)$. Verifying whether it is an encryptions of 1 message boils down to checking the following relations:

$$(\alpha^{s'})^s \;=\; \beta^{s'} .$$

This equality can be verified by $m$ parties each holding distinct shares $s_1, \ldots, s_m$ and $s'_1, \ldots, s'_m$ of the secrets $s$ and $s'$ without reconstructing either secret. This is achieved, assuming participant $j$ commits to her secret shares $s_j$ and $s'_j$ by posting $y_j = g^{s_j}$ and $y'_j = g^{s'_j}$ in her designated area of the bulletin board, by executing three Distributed Exponentiation (DEx) protocols (two for computing $\alpha' = \alpha^{s'}$ and $\beta' = \beta^{s'}$ and the other for verifying that $(\alpha')^s = \beta'$ — in this order). Such protocol on input $\alpha$ outputs $\alpha^s$ by means of the following steps:

1. Participant $j$ posts $\omega_j = \alpha^{s_j}$ and proves in zero knowledge that $\log_g y_j = \log_\alpha \omega_j$ using the Chaum and Pedersen [CP92] protocol for equality of discrete logs described in Appendix A. The protocol is honest–verifier zero–knowledge [CGS97]. This, suffices for our application. In order to make the protocol non–interactive the Fiat–Shamir heuristic is used. This requires a cryptographically strong hash function. We henceforth refer to this non–interactive proof as **Proof–Log**$(g, y_j; \alpha, \omega_j)$.

2. Let $\Lambda$ denote any subset of $t$ participants who successfully passed the proof and let $\lambda_{j,\Lambda} = \prod_{l \in \Lambda \setminus \{j\}} l/(l-j)$ denote the appropriate Lagrange interpolation coefficients. The desired value can be obtained from the following identity:

$$\alpha^s \quad = \quad \prod_{j \in \Lambda} \omega_j^{\lambda_{j,\Lambda}} \, .$$

ElGamal Ballots and Efficient Proofs of Validity: In our MTS each voter will post on the bulletin board an ElGamal encryption. The encryption is accompanied by a proof of validity that shows that the ballot is indeed of the correct form. To implement this, consider a prover who knows $m \in \{m_0, m_1\}$ and wants to show that an ElGamal encryption of $m$, say $(\alpha, \beta) = (g^r, y^r m)$, is indeed of this form without revealing the value of $m$. The prover's task amounts to showing that the following relation holds:

$$\log_g \alpha \quad \in \quad \{\log_y(\beta/m_0), \log_y(\beta/m_1)\} \, ,$$

Building on ideas from [CDS94], an efficient witness indistinguishable (in particular honest–verifier zero–knowledge) proof of knowledge for the above relation was proposed in [CGS97]. For completeness sake we review it as well as a non–interactive version of it in Appendix B. We henceforth refer to this (non–interactive) proof as $\textbf{Proof–Ballot}_{\{m_0,m_1\}}(\alpha, \beta)$.

Universally Verifiable MIX–network: A MIX–network for ElGamal ciphertexts consists of a bulletin board and a collection of authorities called the MIX–servers. It takes a list of ElGamal ciphertexts, permutes them according to some (secret) permutation, and outputs an ElGamal re–encryption of the original list (without ever decrypting the original list of ciphertexts).

We now discuss a proposal by Abe [Abe98] for a MIX–network with the characteristics described above that satisfies the following properties: correctness, robustness, privacy, and universal verifiability. Initially, MIX–servers jointly generate a secret values $s$ and a public value $y$ as in the above mentioned DKG protocol. Moreover, the bulletin board initially contains a list of ElGamal ciphertexts $((G_{0,l}, M_{0,l}))_l$ where $M_{0,l} = m_l y^{t_{0,l}}$ and $G_{0,l} = g^{t_{0,l}}$ for a message $m_l \in G_q$ and $t_{0,l} \in_R \mathbb{Z}_q$. (To avoid the attack shown in [Pfi95] a proof of knowledge of $t_{0,l}$ must accompany $(G_{0,l}, M_{0,l})$.)

The list of encrypted messages is randomized and permuted by the cascade of MIX–servers. Server $j$ chooses a random permutation $\pi_j$ of $S$, for each $l$ the sever picks $t_{j,l} \in_R \mathbb{Z}_q$, reads $((G_{j-1,l}, M_{j-1,l}))_l$ from the bulletin board, and posts in the bulletin board $((G_{j,l}, M_{j,l}))_l$ where

$$G_{j,l} = G_{j-1,\pi_j(l)} g^{t_{j,\pi_j(l)}} \, , \qquad \text{and} \qquad M_{j,l} = M_{j-1,\pi_j(l)} y^{t_{j,\pi_j(l)}} \, .$$

Processing proceeds sequentially through all servers.

The security of the mix performed by servers relies on the following:

**Lemma 1** ([Abe98]) *Under the intractability of the Decision Diffie–Hellman problem, given correctly formed $((G_{j-1,l}, M_{j-1,l}))_l$ and $((G_{j,l}, M_{j,l}))_l$, no adversary can determine $\pi_j(l)$ for any $l$ with probability significantly better than $1/|S|$.*

An additional protocol is executed in order to prove the correctness of randomization and permutation to external verifiers as well as convince honest servers that they have contributed to the output, i.e., no one has canceled the randomization and permutation performed by the honest servers (with success probability significantly better than a random guess). For completeness sake we review this protocol as well as a non–interactive version of it in Appendix C. We henceforth refer to this (non–interactive) proof as **Proof**–$\Pi$.

## 3 Membership Testing Scheme (MTS)

We work in the model described in the previous section where the active set of participants is $V_1, \ldots, V_n$ (the voters) and $A_1, \ldots, A_m$ (the authorities). Voters and authorities might overlap.

In what follows, $N$ denotes the cardinality of the set $S$ for which one seeks to verify whether it contains the vote tally. Also, henceforth, $i$ runs over $\{1, \ldots, n\}$, $j$ runs over $\{1, \ldots, m\}$, and $l$ runs over $S$.

Basic MTS Protocol

Input

1. Public Input: System parameters, i.e., $k$ bit long primes $p$ and $q$, where $q > n$ divides $p - 1$, and generators $g$ and $h$ of a multiplicative subgroup $G_q$ of order $q$ of $\mathbb{Z}_p^*$. A set $S \subset \{1, \ldots, n\}$.
2. Private Input for voter $V_i$: A vote $v_i \in \{0, 1\}$.

Goal

To determine whether $\sum_i v_i$ belongs to $S$ without revealing anything else besides this bit of information.

Setup Phase

1. Using the DKG protocol, $A_1, \ldots, A_m$ jointly generate the public value $y = g^s$ where $s \in_R G_q$ and the private shares $s_1, \ldots, s_m$. Authorities commit to their share $s_j$ of $s$ by posting $y_j = g^{s_j}$ in their designated area of the bulletin board.
2. Using the DKG protocol, authorities jointly generate, for each $l \in S$, the public value $y_l = g^{s'_l}$ where $s'_l \in_R \mathbb{Z}_q$ and the private shares $s'_{l,1}, \ldots, s'_{l,m}$. Authorities commit to their share $s'_{l,j}$ of $s'_l$ by posting $y'_{l,j} = g^{s'_{l,j}}$ in their designated area of the bulletin board.

MIX Phase

1. Let $((G_{0,l}, M_{0,l}))_l$ be a list such that for each $l \in S$,

$$G_{0,l} = 1\,, \quad \text{and} \quad M_{0,l} = h^{-l}\,.$$

2. Authority $A_j$ chooses at random a permutation $\pi_j$ of $\{1, \ldots, N\}$, for each $l$ picks $t_{j,l} \in_R \mathbb{Z}_q$, and posts the list $((G_{j,l}, M_{j,l}))_l$ such that for each $l \in S$,

$$G_{j,l} = G_{j-1,l}\, g^{t_{j,\pi_j(l)}}\,, \quad \text{and} \quad M_{j,l} = M_{j-1,l}\, y^{t_{j,\pi_j(l)}}\,.$$

Verification Phase

Authorities cooperate to issue **Proof–Π** — a honest–verifier zero–knowledge (non–interactive) proof that shows that they know random factors and permutations that relate $((G_{0,l}, M_{0,l}))_l$ with $((G_{m,l}, M_{m,l}))_l$. Each authority signs **Proof–Π** in order to insure verifiers of the presence of an authority they can trust. Each authority checks the proof. If the check succeeds the result is declared VALID. If it fails, dishonest authorities are identified (and removed) by means of the tracing capabilities that **Proof–Π** provides. The remaining authorities restart from the beginning of the MIX Phase.

Voting Phase

Voter $V_i$ chooses $r_i \in_R \mathbb{Z}_q$ and posts an ElGamal encryption representing his vote $v_i$, say $(\alpha_i, \beta_i) = (g^{r_i}, y^{r_i} h^{v_i})$, together with $\mathbf{Proof\text{–}Ballot}_{\{h^0,h^1\}}(\alpha_i, \beta_i)$.

Output Phase

1. Each authority computes $\alpha = \prod_i \alpha_i$, and $\beta = \prod_i \beta_i$.
2. Using the DEx protocol, for each $l \in S$, authorities compute

$$G'_l \;=\; (G_{m,l}\, \alpha)^{s'_l}\ , \quad \text{and} \quad M'_l \;=\; (M_{m,l}\, \beta)^{s'_l}\ .$$

Then, using the DEx protocol again, authorities verify whether for some $l$ in $S$,

$$(G'_l)^s \;=\; M'_l\,.$$

In the affirmative case they output MEMBER, otherwise NON–MEMBER.

**Remark 1** *Note that both the MIX Phase and the Verification Phase may be precomputed before the voting begins. In fact, if the Verification Phase is not declared* VALID*, there is no need to perform the Voting Phase.*

ELECTRONIC JURY VOTING PROTOCOL: We conclude this section with a simple observation; an electronic analog of a 12–juror civil case voting protocol where 9 votes suffice to reach a verdict can be derived from our Basic MTS by letting $n = 12$ and $S = \{9, 10, 11, 13\}$.

## 4 Analysis

ELIGIBILITY: The non–anonymity of ballot casting insures that only authorized voters cast ballots. Indeed, recall that voters must identify themselves through digital signatures in order to post their vote onto their designated area of the bulletin board. This also insures that no voter can cast more than one ballot.

NO–DUPLICATION: Follows from the voter specific challenge in the (non–interactive) proof of validity of ballots. Indeed, recall that each voter computes the challenge in the proof of validity of ballots described in Appendix B as a hash of several values one of which is a unique public key identifying the voter.

CORRECTNESS: Clearly, an honest voter can construct a ballot and its accompanying proof of validity. Moreover, the following holds:

**Theorem 1** *If all participating authorities are honest, then they will output MEMBER if and only if the tally of the validly cast votes belongs to the set $S$.*

**Proof:** In addition to the notation introduced in Sect. 3, for $j \leq m$ let $\pi_{j,\ldots,m}$ denote $\pi_j \circ \pi_{j+1} \circ \ldots \circ \pi_m$ and $\pi$ denote $\pi_1 \circ \ldots \circ \pi_m$. Assuming that all participants in the Basic MTS are honest,

$$G_{m,l} = g^{\tau_{m,l}}, \quad \text{and} \quad M_{m,l} = y^{\tau_{m,l}} h^{-\pi(l)},$$

where

$$\tau_{m,l} = \sum_j t_{j,\pi_{j,\ldots,m}(l)}, \quad \text{and} \quad \rho = \sum_i r_i .$$

Hence, since $y = g^s$, $\alpha = g^\rho$, and $\beta = y^\rho h^{\sum_i v_i}$,

$$(G'_l)^s = y^{s'_l(\tau_{m,l}+\rho)}, \quad \text{and} \quad M'_l = y^{s'_l(\tau_{m,l}+\rho)} \cdot h^{s'_l(\sum_i v_i - \pi(l))} . \tag{1}$$

If $\sum_i v_i \in S$, for $l = \pi^{-1}(\sum_i v_i)$, it holds that $\pi(l) = \sum_i v_i$. Hence, $h^{s'_l(\sum_i v_i - \pi(l))} = 1$, and the LHS of both equalities in (1) are equal. If $\sum_i v_i \notin S$, then for every $l \in S$ it holds that $\pi(l) \neq \sum_i v_i$. Hence, $h^{s'_l(\sum_i v_i - \pi(l))} \neq 1$, and the LHS of both equalities in (1) are distinct for every $l$ in $S$. Thus, the Basic MTS outputs MEMBER if and only if $\sum_i v_i$ belongs to $S$. ∎

ROBUSTNESS: First we observe that robustness with respect to malicious voters is achieved.

**Lemma 2** ([CGS97])*) An incorrectly formed ballot will be detected with overwhelming probability.*

Still, we need to show that the protocol cannot be disrupted by dishonest authorities. We will need the following:

**Lemma 3** ([Abe98]) Protocol–$\Pi$ *is a honest verifier zero–knowledge proof of knowledge for $\pi$ and $\tau_{m,l}$'s. The protocol is also honest verifier zero–knowledge proof of knowledge for $\pi_j$'s and $t_{j,l}$'s held by honest provers.*

Robustness with respect to malicious authorities is now guaranteed by the following:

**Theorem 2** *Assume there are at most $m - t - 1$ participating authorities controlled by an adversary. The goal of the adversary is to force the output of the scheme to be incorrect (i.e., to be MEMBER when it should be NON–MEMBER and vice versa). The adversary cannot succeed with non–negligible probability, and the identity of the authorities controlled by the adversary will be learned with overwhelming probability.*

**Proof: (Sketch)** By Lemma 2 it suffices to consider the case were only correctly formed ballots will be accounted for. Let $T$ be the tally of the correctly formed ballots.

Observe that there are at least $t$ honest authorities. Any such collection of authorities will be able to decide correctly whether or not $T$ belongs to $S$ unless $((G_{m,l}, M_{m,l}))_l$ is not a permuted ElGamal re–encryption of $((G_{0,l}, M_{0,l}))_l$. If the latter holds, then Lemma 3 insures that with overwhelming probability the Verification Phase will detect it, the tracing option invoked, and the identity of dishonest authorities exposed. ∎

PRIVACY: We now show that under a standard computational assumption our Basic MTS does not disclose any information pertaining the honest voter's ballots besides that implied by the output of the scheme. Specifically, the following holds.

**Theorem 3** *Assume there are less than $t$ dishonest authorities and $n'$ dishonest voters controlled by an adversary. Let $T_h$ and $T_d$ be the tally of the correctly emitted ballots among the honest and dishonest voters, respectively. The goal of the adversary is to learn any additional information concerning the votes cast by honest voters, besides that implied by whether or not $T_h$ belongs to $(S - T_d) \cap \{0, \ldots, n - n'\}$.*[2] *Under the Diffie–Hellman assumption, the adversary has a negligible probability of success.*

**Proof: (Sketch)** By Lemma 2 it suffices to consider the case were only correctly formed ballots will be accounted for.

If an adversary succeeds in learning any additional information besides that implied by whether or not $T_h$ belongs to $(S - T_d) \cap \{0, \ldots, n - n'\}$, then such an adversary:

(i) either does not learn any information concerning the tally of correctly emitted ballots $T_h$ besides that implied by whether or not $T_h$ belongs to $(S - T_d) \cap \{0, \ldots, n - n'\}$, but succeeds in learning some information concerning the votes of honest voters besides that implied by membership of $T_h$ in $(S - T_d) \cap \{0, \ldots, n - n'\}$, or,

(ii) succeeds in learning some information concerning the tally $T_h$ of correctly emitted ballots besides that implied by membership of $T_h$ in $(S - T_d) \cap \{0, \ldots, n - n'\}$.

In case (i), a reduction type argument based on the the existence of such an adversarial procedure implies that the underlying homomorphic electronic election scheme of Cramer et al. is not private. This would contradict the assumption on which Cramer et al.'s proposal relies, i.e., the Diffie–Hellman assumption [CGS97, Theorem 2].

We now consider case (ii). Let $T$ denote the tally and $X_l$ denote the random variable corresponding to the value of the element of $S$ encoded by $(G_{m,l}, M_{m,l})$. We model any prior knowledge about the tally as a known distribution, i.e., we think of $T$ as a random variable of known distribution.

By the end of the Output Phase some information concerning $T$ is revealed. Indeed, in case $T \notin S$, it is determined that $X_l \neq T$ for every $l$, and in case $T \in S$, that $X_{l^*} = T$ for an $l^*$ and that $X_l \neq T$ for all $l \neq l^*$. Given that $s'_l$ is uniformly distributed over $\mathbb{Z}_q$ and is independent of $(s'_{l'})_{l' \neq l}$, one has that $(G'_l)^s / M'_l$ is uniformly distributed over $G_q$ if $X_l \neq T$. Moreover, $(G'_l)^s / M'_l$ and $((G'_{l'})^s / M'_{l'})_{l' \neq l}$ are independently distributed. Thus, at the end of the Output Phase the only information revealed is, in case $T \notin S$, that $X_l \neq T$ for every $l$, and in case $T \in S$, that $X_{l^*} = T$ for an $l^*$ and that $X_l \neq T$ for all $l \neq l^*$.

We claim that at the end of a successful run of the protocol the only information disclosed concerning $X_l$'s is given by the a posteriori distributions $X_l/(T \in S)$ in case $T \in S$, and $X_l/(T \notin S)$ in case $T \notin S$. Indeed, we know from Theorem 2 that as long as $t$ authorities are honest, the computation is robust. Thus, if a result is declared VALID, then it must be so with an overwhelming probability. This guarantees that the permutation and randomization performed by honest authorities during the MIX Phase was not canceled. So, for any permutation $(s_l)_{l \in S}$ of $S$ and arbitrary $i$,

$$\mathbb{P}\left[\, (X_l)_{l \in S} = (s_l)_{l \in S}, T = i \,\right] \;=\; \frac{1}{|S|!} \cdot \mathbb{P}\left[\, T = i \,\right] ,$$

where probabilities are taken over the random permutation performed during the MIX Phase and any prior knowledge about $T$. It immediately follows that for any $i$,

$$\mathbb{P}\left[\, T = i | X_l \neq T, \forall l \in S \,\right] \;=\; \mathbb{P}\left[\, T = i | T \notin S \,\right] ,$$

[2] For a set of integers $\Omega$ and an integer $x$ the set $\{\omega - x : \omega \in \Omega\}$ is denoted by $\Omega - x$.

and,

$$\mathbb{P}\left[T = i | X_{l^*} = T; X_l \neq T, \forall l \in S \setminus \{l^*\}\right] \;=\; \mathbb{P}\left[T = i | T \in S\right].$$

Hence, if during the MIX Phase the adversary can obtain any information besides that implied by whether $T$ belongs or not to $S$, then a reduction type argument implies that there is an efficient procedure that correlates input elements to output elements of the honest authorities MIX Phase computation. This would contradict the intractability assumption on which the underlying MIX network's security claim relies, i.e., the decision Diffie–Hellman problem.

If the result of the Verification Phase is not declared VALID, then honest authorities will perform the tracing of dishonest authorities. Since both the MIX Phase and Verification Phase are independent of the votes cast, this does not disclose any information that allows the adversary to compromise the voters' privacy (even if the Voting Phase is performed, see Remark 1). In case the Verification Phase is not declared VALID, then the Output Phase will not be performed and thus no information concerning $T_h$ would be leaked (besides that which can be obtained from the ElGamal encryption of $T_h$). ∎

Universal Verifiability: Follows from the public verifiability of the proofs of ballot validity (**Proof–Ballot**), the proof of randomization and permutation (**Proof**–$\Pi$), and the proof of knowledge of equality of discrete logarithms (**Proof–Log**).

Efficiency: We make the (realistic) assumption that $n \geq N \geq t$. Recall that a modular multiplication of $O(k)$ bit sized numbers is our unit measure of computational costs.

The voter's ballot consists of an ElGamal ciphertext and a (non–interactive) proof that it is indeed a valid ballot. The size of both components is linear in the size of an element of $\mathbb{Z}_p^*$, i.e., $O(k)$. The computational work involved in the computation of both ballot components is dominated by the modular exponentiations, of which there are a constant number, each one requiring $O(k)$ work. Hence, the computational and communicational complexity of the voter's protocol is linear in the security parameter $k$ — thus optimal. Moreover, for any reasonable security parameter, a voter's protocol remains the same even if the number of voters varies. The work needed to verify that a voter sent in a well formed ballot equals the computational cost of making the ballot, i.e., $O(k)$ per voter. We stress that all the above characteristics of a voter's protocol are inherited from the electronic election scheme proposed in [CGS97].

The work performed by the $j$–th authority during the MIX Phase is dominated by the cost of computing $((G_{j,l}, M_{j,l}))_l$. Since

$$G_{j,l} \;=\; G_{j-1,l}\, g^{t_{j,\pi_j(l)}}, \qquad \text{and} \qquad M_{j,l} \;=\; M_{j-1,l}\, y^{t_{j,\pi_j(l)}},$$

the work performed by each authority during this phase is $O(Nk)$. Analogously, the work performed by each authority during the Verification Phase is $O(Nk^2)$. Finally, since each run of the DEx protocol costs $O(mk)$ per authority, the work performed by each authority during the Output Phase is $O((mN+n)k)$ (the $O(nk)$ term is due to the work performed in order to compute $\alpha = \prod_i \alpha_i$, and $\beta = \prod_i \beta_i$). The other tasks performed by the authorities are not relevant in terms of computational costs. Thus, the work performed by each authority is $O((mN+n)k)$ provided they spend $O(Nk^2)$ work during pre-computation. The communicational complexity (in bits) incurred by each authority exceeds the computational complexity by a factor of $k$.

The computational complexity of verifying the authorities work is proportional to the computational work performed by each authority during the corresponding phase.

## 5 Variants

MORE EFFICIENT AND ALTERNATIVE MTSs: If one is willing to forgo universal verifiability, then more efficient MIX–networks like the one proposed by Jakobsson and Desmedt and Kurosawa [Jak98, DK00] might be used instead of Abe's MIX–network in the MTS of Sect. 3. In this case, the work done by each authority during the pre-computation stage is reduced to $O((m + k/\log N)Nk)$. In fact, the only essential characteristic our MTS scheme requires from the underlying MIX–network is that it performs a random secret permutation and ElGamal re–encryption of an input list of ElGamal ciphertexts. (The threshold decryption capabilities utilized in the DEx protocol is a feature from the underlying encryption scheme, not of the MIX–network). Thus, other recent MIX–network proposals like those of Jakobsson et al. [Jak99, JJ99] are good candidates for drop in replacements in the MIX module of the MTS of Sect. 3.

RECEIPT–FREE MTSs: In some situations it might be desirable to ensure that parties involved in the MTS can not sell or be coerced into revealing the values they cast. For example, in electronic jury voting one desires voters can neither sell nor be coerced into disclosing their votes, i.e., the scheme should be receipt–free. Our basic MTS proposal can be enhanced in order to achieve this property. Indeed, it suffices to use Hirt and Sako's [HS00] electronic election scheme instead of the one of of Cramer et al. [CGS97]. Note that this requires new assumptions, among them, physical assumptions like the existence of secret one–way communication channels from the authorities to the voters. Moreover, the computational work as well as the communication complexity of Hirt and Sako's scheme increases linearly with the number of parties/voters involved.

## 6 Applications

TESTING MEMBERSHIP OF LINEAR FUNCTIONS: We can modify our Basic MTS to allow parties $P_1, \ldots, P_n$ to determine whether their private inputs $v_i \in S_i$, for $i \in \{1, \ldots, n\}$, are such that $\sum_i c_i v_i \in S$ without revealing $\sum_i c_i v_i$. Here, $S_1, \ldots, S_n$ and $S$ are publicly known subsets of $\mathbb{Z}_q$, and $c_1, \ldots, c_n$ is a publicly available fixed sequence of integers. This modification of our Basic MTS allows to implement a weighted majority voting electronic election scheme.

SCORING: Consider a person/entity which is willing to answer $n$ very sensitive questions to a group of $m$ evaluators (say for a job interview, insurance application, etc). Assume the $i$–th question accepts as answer any element of $S_i$. Each evaluator would like to learn whether the weighted score of the answers $\sum_i c_i a_i$ exceeds a threshold (here again $c_1, \ldots, c_n$ is a publicly available fixed sequence of integers). But, the respondent wishes to keep private the answers to each individual question. This problem clearly reduces to the one discussed in the previous paragraph. Thus, it follows that our Basic MTS can be used to solve it.

## 7 Final Comments

An interesting feature of our jury voting scheme is that it combines two parallel lines of research concerning electronic voting schemes, one based on MIX–networks [Cha81] and the other on homomorphic encryptions [CF85, BY86, Ben87]. We need homomorphic encryption in order to hide the ballots content and compute the tally while keeping it secret. We need ElGamal based MIX–networks in order to hide the value of the elements of $S$ to which the ElGamal encryption of the vote tally is compared. It is an interesting challenge to design an electronic jury voting scheme in the model introduced in [CF85, BY86, Ben87] which does not rely on MIX–networks.

## Acknowledgement

We would like to thank Mihir Bellare, Daniele Micciancio, and Rosario Gennaro for interesting and helpful discussions, Martin Loebl for suggesting the SCORING application, and Valtteri Niemi for making his work available to us.

## References

[Abe98] M. Abe. Universally verifiable MIX-net with verification work independent of the number of MIX-servers. In *EuroCrypt'98*, 437–447. Springer-Verlag. LNCS Vol. 1403.

[Ben87] J. Benaloh. *Verifiable Secret–Ballot Elections.* PhD thesis, Yale University, Dept. of Computer Science, New Haven, CT, Sep 1987.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, 1–10, 1988.

[Boy89] C. Boyd. A new multiple key cipher and an improved voting scheme. In *EuroCrypt'89*, 617–625. Springer-Verlag. LNCS Vol. 434.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, 544–553, 23–25 1994.

[BY86] J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing - PODC'86*, 52–62. ACM.

[CCD87] D. Chaum, C. Crépeau, and I. B. Damgård. Multiparty unconditionally secure protocols. In *Crypto'87*, 462–462. Springer-Verlag. LNCS Vol. 293.

[CF85] Josh D. Cohen and Michael J. Fischer. A robust and verifiable cryptographically secure election scheme. In *26th Annual Symposium on Foundations of Computer Science*, 372–382, 1985. IEEE.

[CFSY96] R. Cramer, M. K. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret-ballot elections with linear work. In *EuroCrypt'96*, 72–83. Springer-Verlag. LNCS Vol. 1070.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EuroCrypt'97*, 103–118. Springer-Verlag. LNCS Vol. 1233.

[Cha81] D. L. 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 *EuroCrypt'88*, 177–182. Springer-Verlag. LNCS Vol. 330.

[CP92] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *Crypto'92*, 89–105. Springer-Verlag. LNCS Vol. 740.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge simplified design of witness. In *Crypto'94*, 174–187. Springer-Verlag. LNCS Vol. 839.

[DK00] Y. Desmedt and K. Kurosawa. How to Break a Practical MIX and Design a New One. In *EuroCrypt'00*, 557–572. Springer-Verlag. LNCS Vol. 1807.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions of Information Theory*, IT-31(4):469–472, July 1985.

[FOO92] A. Fujioka, T. Okamoto, and K. Ohta. A practical digital multisignature scheme based on discrete logarithms. In *AusCrypt'92*, 244–251, 1992. Springer-Verlag. LNCS Vol. 718.

[Gen95a] R. Gennaro. A receipt–free election scheme tolerating a dynamic coercer (with applications to key escrow). Private Communication, Nov 1995.

[Gen95b] R. Gennaro. Achieving independence efficiently and securely. In *Proc. 14th ACM Symposium on Principles of Distributed Computing - PODC'95*. ACM.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *EuroCrypt'99*, 295–310. Springer-Verlag. LNCS Vol. 1592.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th Annual Symposium on Foundations of Computer Science*, 174–187, 1986. IEEE.

[HS00] M. Hirt and K. Sako. Efficient Receipt–Free Voting Based on Homomorphic Encryption. In *EuroCrypt'00*, 539–556. Springer-Verlag. LNCS Vol. 1807.

[Ive91] K. R. Iversen. A cryptographic scheme for computerized general elections. In *Crypto'91*, 405–419. Springer-Verlag. LNCS Vol. 576.

[Jak98] M. Jakobsson. A practical mix. In *EuroCrypt'98*, 448–461. Springer-Verlag. LNCS Vol. 1403.

[Jak99] M. Jakobsson. Flash mixing. In *Proc. 18th ACM Symposium on Principles of Distributed Computing - PODC'99*, 83–89. ACM.

[JJ99] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Tech. Rep. 99–33, DIMACS, 1999.

[NR94] V. Niemi and A. Renvall. How to prevent buying of votes in computer elections. In *AsiaCrypt'94*, 164–170, 1994. Springer-Verlag. LNCS Vol. 917.

[OO89] T. Okamoto and K. Ohta. Divertible zeroknowledge interactive proofs and commutative random self-reducibility. In *EuroCrypt'89*, 134–149. Springer-Verlag. LNCS Vol. 434.

[Ped91] T. P. Pedersen. Distributed provers with applications to undeniable signatures. In *EuroCrypt'91*, 221–242. Springer-Verlag. LNCS Vol. 547.

[Pfi95] B. Pfitzmann. Breaking an efficient anonymous channel. In *EuroCrypt'94*, 332–340. Springer-Verlag. LNCS Vol. 950.

[PIK93] C. Park, K. Itoh, and K. Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *EuroCrypt'93*, 248–259. Springer-Verlag. LNCS Vol. 765.

[Sch99] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Crypto'99*, 148–164. Springer-Verlag. LNCS Vol. 1666.

[SGR97] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, 44–54, IEEE, 1997.

[SK94] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *Crypto'94*, 411–424. Springer-Verlag. LNCS Vol. 839.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme — A practical solution to the implementation of a voting booth. In *EuroCrypt'95*, 393–403. Springer-Verlag. LNCS Vol. 921.

# A Proof of Knowledge for Equality of Discrete Logarithms

In order to make the protocol of Fig. 1 non–interactive the Fiat–Shamir heuristic is used. This maintains security in the random oracle model.

$$
\begin{array}{ccc}
\textsf{Prover} & & \textsf{Verifier} \\
[(y_j, \omega_j) = (g^{s_j}, \alpha^{s_j})] & & \\
\rho \in_R \mathbb{Z}_q & & \\
(a, b) \leftarrow (g^\rho, \alpha^\rho) & \xrightarrow{a,b} & \\
 & \xleftarrow{c} & c \in_R \mathbb{Z}_q \\
r \leftarrow \rho + s_j c & \xrightarrow{r} & g^r \stackrel{?}{=} a y_j^c \\
 & & \alpha^r \stackrel{?}{=} b \omega_j^c
\end{array}
$$

Figure 1: Proof of knowledge for $\log_g y_j = \log_\alpha \omega_j$.

$$
\begin{array}{l|lcl}
\multicolumn{2}{c}{\textsf{Voter}} & & \textsf{Verifier} \\
\multicolumn{1}{c}{v = 0} & \multicolumn{1}{c}{v = 1} & & \\
\hline
\rho, w, r_1, d_1 \in_R \mathbb{Z}_q & \rho, w, r_2, d_2 \in_R \mathbb{Z}_q & & \\
 & & & \\
\alpha \leftarrow g^\rho & \alpha \leftarrow g^\rho & & \\
\beta \leftarrow y^\rho & \beta \leftarrow y^\rho h & & \\
a_1 \leftarrow g^{r_1} \alpha^{d_1} & a_1 \leftarrow g^w & & \\
b_1 \leftarrow y^{r_1} (\beta/h)^{d_1} & b_1 \leftarrow y^w & & \\
a_2 \leftarrow g^w & a_2 \leftarrow g^{r_2} \alpha^{d_2} & & \\
b_2 \leftarrow y^w & b_2 \leftarrow y^{r_2} \beta^{d_2} & \xrightarrow{\alpha,\beta,a_1,b_1,a_2,b_2} & \\
d_2 \leftarrow c - d_1 & d_1 \leftarrow c - d_2 & \xleftarrow{c} & c \in_R \mathbb{Z}_q \\
r_2 \leftarrow w - \rho d_2 & r_1 \leftarrow w - \rho d_1 & \xrightarrow{d_1,d_2,r_1,r_2} & c \stackrel{?}{=} d_1 + d_2 \\
 & & & a_1 \stackrel{?}{=} g^{r_1} \alpha^{d_1} \\
 & & & b_1 \stackrel{?}{=} y^{r_1} (\beta/h)^{d_1} \\
 & & & a_2 \stackrel{?}{=} g^{r_2} \alpha^{d_2} \\
 & & & b_2 \stackrel{?}{=} y^{r_2} \beta^{d_2}
\end{array}
$$

Figure 2: Proof of Validity of Encrypted vote $(\alpha, \beta) = (g^\rho, y^\rho h^v), \rho \in_R \mathbb{Z}_q, v \in \{0, 1\}$

# B Proof of Validity of Ballot

In order to make the protocol of Fig. 2 non–interactive, the challenge $c$ is computed as a hash of the values $\alpha$, $\beta$, $a_1$, $b_1$, $a_2$, $b_2$. This maintains security in the random oracle model. In order to prevent vote duplication, the challenge must be made voter–specific. Following [Gen95b] this can be done by having voter $V_i$ compute the challenge as the hash of the values $ID_i$, $\alpha$, $\beta$, $a_1$, $b_1$, $a_2$, $b_2$, where $ID_i$ is a unique public string identifying $V_i$.

# C Proof of Randomization and Permutation

We now describe a protocol due to Abe [Abe98], denoted **Protocol**–$\Pi$, which convinces external verifiers of the correctness of the randomization and permutation. It also convinces honest servers that the permutations they have performed have not been canceled (with non–negligible probability).

PROTOCOL–II

The following steps are repeated $k$ times.

1. The $j$-th server receives $((\widetilde{G}_{j-1,l}, \widetilde{M}_{j-1,l}))_l$. She then selects a random permutation $\widetilde{\pi}_j$ of $\{1,\ldots,N\}$ and sends $((\widetilde{G}_{j,l}, \widetilde{M}_{j,l}))_l$ to the $(j+1)$–th server, where
$$\widetilde{G}_{j,l} = \widetilde{G}_{j-1,\widetilde{\pi}_j(l)} g^{\widetilde{r}_{j,\widetilde{\pi}_j(l)}}, \quad \text{and} \quad \widetilde{M}_{j,l} = \widetilde{M}_{j-1,\widetilde{\pi}_j(l)} y^{\widetilde{r}_{j,\widetilde{\pi}_j(l)}},$$
where $\widetilde{r}_{j,l} \in_R \mathbb{Z}_q$. The last server publishes the list she receives.

2. Verifier publishes the challenge $c \in_R \{0,1\}$.

3. If $c = 0$, the $j$–th server commits to $j$, $\widetilde{\pi}_j$, $\widetilde{r}_{j,1},\ldots,\widetilde{r}_{j,N}$ by broadcasting to all other servers a hash of these values. Commitments are opened once all of them have been exchanged. The last server posts $\widetilde{\pi} = \widetilde{\pi}_1 \circ \ldots \circ \widetilde{\pi}_m$ and $\widetilde{\rho}_{m,l} = \sum_j \widetilde{r}_{j,\widetilde{\pi}_j(l)}$.

   If $c = 1$, the $j$–th server calculates $\varphi_j = \pi_j^{-1} \circ \varphi_{j-1} \circ \widetilde{\pi}_j$ and $\widetilde{\omega}_{j,l} = \widetilde{\omega}_{j-1,\widetilde{\pi}_j(l)} + \widetilde{r}_{j,\widetilde{\pi}_j(l)} - t_{j,\varphi_{j-1}\circ\widetilde{\pi}_j(l)}$ (for $j = 0$, $\varphi_0$ equals the identity permutation, and $\widetilde{\omega}_{0,l} = 0$). The last server publishes $\varphi = \varphi_m$ and all $\widetilde{\omega}_{m,l}$'s.

4. If $c = 0$ each server and verifier check that all commitments where opened correctly, that $\widetilde{\pi}$ and $\widetilde{\rho}_{m,l}$'s are correctly made, and whether
$$\frac{\widetilde{G}_{m,l}}{G_{0,\widetilde{\pi}(l)}} = g^{\widetilde{\rho}_{m,l}}, \quad \text{and} \quad \frac{\widetilde{M}_{m,l}}{M_{0,\widetilde{\pi}(l)}} = y^{\widetilde{\rho}_{m,l}},$$
and in case $c = 1$,
$$\frac{\widetilde{G}_{m,l}}{G_{m,\varphi(l)}} = g^{\widetilde{\omega}_{m,l}}, \quad \text{and} \quad \frac{\widetilde{M}_{m,l}}{M_{m,\varphi(l)}} = y^{\widetilde{\omega}_{m,l}}.$$
   If the verification fails, the randomization and permutation step is declared not VALID and all servers show the $\pi_j$'s and $t_{i,j}$'s. This makes all computations traceable and dishonest servers are identified.

Although the above protocol is interactive, a non–interactive version can be derived using the Fiat–Shamir heuristic by computing the challenge via a hash function. In this case, the non–interactive proof consists of the outputs of the last server.