An extended abstract of this paper appeared in the *Proceedings of 5-th Latin American Symposium on Theoretical Informatics* - LATIN'02, Lecture Notes in Computer Science Vol. 2286, S. Rajsbaum ed., Springer–Verlag, 2002. This is full version.

Electronic Jury Voting Protocols

Alejandro Hevia* Marcos Kiwi†

June 24, 2003

Abstract

This work stresses 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 small 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.

*Dept. of Computer Science & Engineering, University of California at San Diego, Mail Code 0114, 9500 Gilman Drive, La Jolla, California 92093, USA, and Dept. Cs. de la Computación, U. Chile. E-mail: ahevia@cs.ucsd.edu, URL: http://www-cse.ucsd.edu/users/ahevia. Partially supported by Conicyt via Fondap in Applied Mathematics 1999–2000, Fondecyt No. 1981182, NSF CCR-0093029, and Mideplan Scholarship.

†Dept. Ing. Matemática, U. Chile & Ctr. de Modelamiento Matemático, UMR 2071 U. Chile–CNRS, Santiago 170-3, Chile. E-mail: mkiwi@dim.uchile.cl. Gratefully acknowledges the support of Conicyt via Fondecyt No. 1981182 and Fondap in Applied Mathematics 1999–2000.

Contents

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 follow-the-pack type conducts 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.

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 whenever small groups wish to make a yes/no type decision by majority vote, 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, whether or not to send an invitation to a speaker, to decide whether to go forth with a given investment.

Our main procedure also provides a novel solution for the problem of computing partial information from private data, which includes among others, the 'scoring' problem. In the latter, a person is willing to answer some very sensitive questions to a group of evaluators (say for a job interview or insurance application). Answers are coded as integer values and might be weighted differently depending on the question. Evaluators would like to learn whether the weighted score of the answers T exceeds a given threshold or belongs to a set S of "satisfactory" values. The respondent wishes to keep private the answers to each individual question. A solution satisfying both requirements can be obtained by using a threshold voting scheme. Here, answers to different questions are seen as votes coming from different individuals. The (weighted) sum T of these "votes"is tested for membership in the set S of "satisfactory" values. This work's main scheme provides a solution to this problem and guarantees that only one bit of information is released: whether the "tally" T belongs or not to the given set S.

The first electronic voting scheme proposals focused on breaking the correspondence between the

voters and the vote casted. Afterward, 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 other applications.

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 built, can be obtained. Indeed, the application of general techniques like the ones proposed in [33, 14], and [8] yield such constructions. Unfortunately, the solutions thus obtained do not exhibit some of the properties one desires of an electronic voting scheme (e.g., non–interaction among voters). On the contrary, homomorphic voting protocols, MIX–network based protocols, and verifiable secret sharing protocols are, in general, more efficient and require less communication than general purpose secure multi-party computation protocols.

Electronic voting schemes are one of the prime examples of secure multi–party computation. This partly explains why they have been intensively studied. The first electronic election scheme in the literature was proposed by Chaum [21]. 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 work, several other electronic election schemes based on untraceability networks have been proposed. Among the earlier ones are [22, 12, 50, 54, 27]. More recent proposals of these type are those of [44, 40, 9, 34]. Improving the efficiency and robustness of MIX-networks has also been a major focus of attention [45, 36, 1, 37, 2, 25, 46, 39, 3, 28, 44, 40, 34]. (For actual implementations of MIX–networks see [52] and the references therein.)

In contrast to the above mentioned schemes [17, 7, 5] introduced ones that 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 [53, 6]. More recent proposals are [18, 19, 51, 35, 4].

The problem of testing whether some value belongs to a predefined set without revealing the tested value or the content of the set, is related to performing "blind queries on databases", or private information retrieval [15, 29]. Indeed, in the case of a single voter with vote v, and assuming the contents of set S are not known to the voter, a procedure that answers whether or not v is in set S, yields a method of searching for v on the "database" S without revealing neither the target value v nor the contents of the database. The reader is referred to [42] for an in-depth survey on this topic.

Recently, Boudot [11] proposed a scheme to securely prove that a committed number T lies in a specific interval $[a, b]$. Although efficient, it strongly relies on the prover's knowledge of T. However, in the jury setting there is no single entity allowed to know the tally T. Hence, as presented and without resorting to secure multi-party computation techniques, Boudot's scheme does not

immediately provide a solution to the jury voting problem. In contrast, our proposed scheme can handle not only interval sets, but also arbitrary sets.

This paper's proposal crucially relies on the feasibility of checking whether two ciphertexts encode the same plaintext, *without* any party performing a decryption. A procedure for efficiently performing such equality test, for ElGamal type ciphertexts, has also been used in [20, 38, 10]. But, it at least dates back to [30].

1.2 Our Contributions

This work's first contribution is that it stresses 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 small 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 type case). For the sake of simplicity of exposition, we 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. Our main MTS satisfy the following properties:

- Eligibility: Only authorized voters can vote and none more than once.
- Correctness: If all participants are honest, the correct output is 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, even 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.

We also exhibit variants to our main MTS that establish trade-offs between some of the above listed features and efficiency of the overall scheme.

In our scheme the voters send in a ballot identical to those proposed in [19], i.e., an ElGamal ciphertext representing his/her vote plus a proof that the ciphertext is indeed a valid ballot. Hence, as in [19], both the computational and communication complexity of the voter's protocol is linear in the security parameter k — thus optimal.[1] Moreover, for any reasonable security parameter, the

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

voters' protocol remains the same even if the number of voters varies. Assuming m authorities, the work performed by each authority is $O(((m+k)|S|+n)k)$. Moreover, the computational complexity of verifying each authority's work is proportional to the work performed by each authority. As in [19] the work needed to verify that a voter sent in a well formed ballot is $O(k)$ per voter.

Our MTS proposal combines features of two parallel lines of research concerning electronic voting schemes, those based on MIX–networks (a la [21]) and in homomorphic encryption schemes (a la [17, 7, 5]). We use homomorphic (ElGamal) encryption in order to hide the vote tallies. Specifically, our MTS relies on the homomorphic properties of ElGamal to construct a ciphertext of the vote tallies. Moreover, it uses MIX–networks (ElGamal based) in order to hide the value of the members of S. This is achieved by generating a randomly permuted list of ciphertexts of elements in S. We rely on special properties of the ElGamal cryptosystem in order to perform an equality test between the tally's ciphertext and each of the ciphertexts corresponding to an element of S. Thus, our MTS proposal crucially relies on the possibility of testing whether two ElGamal ciphertexts correspond the the same plaintext, *without* any of the parties involved performing a decryption.

To the best of our knowledge, the only other cryptographic protocols which rely both on homomorphic encryption schemes and MIX–networks are the independent recent proposals of Hirt and Sako [35] and Jakobsson and Juels [38]. But, our MTS combines both theses schemes in a novel way. Indeed, Hirt and Sako's proposal uses a MIX–network in order to randomly permute, for each voter, potential ballots, while Jakobsson and Juels' scheme permutes truth tables rows to compute the output of each Boolean gate of a circuit. In contrast, 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. By combining MIX–networks with efficient and available cryptographic protocols (namely, verifiable secret sharing and homomorphic voting), this paper gives a first (practical) solution to the mentioned jury voting problem that does not rely on general secure multi-party computation techniques.

We propose several implementations of a MTS. Our first proposal relies on the homomorphic encryption based electronic election scheme of Cramer, Gennaro and Schoenmakers [19] and the MIX–network of Abe [1]. We also discuss alternative implementations of our MTS based on the MIX–network proposals of [36, 25, 28, 44, 40, 34] as opposed to that of [1]. Our different MTS implementations exhibit different properties depending on the previous work we use to build them.

Organization: In Section 2, we informally outline the protocol and discuss the building blocks on which our basic MTS proposal relies. In Section 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 whether the final tally exceeds or not a given threshold. In Section 5 and Section 6 we discuss variants and other applications of our basic scheme. We conclude in Section 7 discussing a feature of all of the MTSs that we propose and some desirable future developments.

2 Preliminaries

We work in the model introduced by Benaloh et al. (see [17, 7, 5] and [19]), 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 polynomially–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 values: a k bit long prime p, a large prime q such that q divides $p-1$, and generators g and h of an order q multiplicative subgroup G_q of $\mathbb{Z}_p^*$. The generators g and h are assumed to be uniformly and independently chosen among the generators of G_q. One way for participants to collectively generate these system values is to run the same probabilistic algorithm over jointly generated uniformly and independent coinflips.

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 Ω. Furthermore, negligible and overwhelming probability correspond to probabilities that are at most $\nu(k)$ and at least $1-\nu(k)$ 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 Protocol Overview

The protocol consists of five main stages. (Setup, MIX, Verification, Voting and Output). In the setup phase, shared parameters subsequently used in the protocol are selected. In the mix phase, the list of encryptions of the elements in a fixed set S is shuffled by a MIX–network. To shuffle the list means permuting it while re–randomizing each of its entries. Hence, the MIX-network's output is a randomly permuted list of re–encryptions of elements in S. Next, in the voting stage, each voter posts an encryption of his vote (using the authorities' jointly generated public–key) and a publicly verifiable proof that the encryption corresponds to a valid vote [19]. Using the homomorphic property of the underlying encryption scheme the authorities proceed to compute the encryption of the tally. Finally, in the output stage, each element of the MIX–network's output list is compared with the encryption of the tally to test whether they encrypt the same plaintext. This stage is performed in such a way that the authorities do not actually decrypt the tally nor the encryption of any element in the shuffled list. Moreover, no information concerning any of the plaintexts involved is revealed. Instead a "blinded" copy of the difference between the tally and each element in the shuffled list is implicitly decrypted. The protocol relies on the fact that discrete exponentiation is injective to unequivocally identify an encryption of 0, and therefore, when two encrypted values are the same. The verification stage checks whether all previous phases were correctly performed.

2.2 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. 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 [19] 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 hold:

- **Correctness:** There is an efficient procedure that on at least $t+1$ shares submitted by honest parties and the public values produced by the DKG protocol, outputs the unique secret value s, even if up to t shares come from faulty parties. 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 formal definition in terms of simulatability see [32]).

The first DKG protocol was proposed by Pedersen [48]. Henceforth in this work, DKG refers to the protocol presented in [32] 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 [26] proposed in [19]. Recall that in ElGamal's cryptosystem $x \in G_q$ is encrypted as $(\alpha, \beta) = (g^r, y^r x)$ for $r \in_R \mathbb{Z}_q$, 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. We now recall the discussion in [30] on how to perform such check efficiently. Indeed, assume (α, β) is an ElGamal encryption of message x, that is $(\alpha, \beta) = (g^r, y^r x)$. Verifying whether it is an encryption of $x = 1$ boils down to checking if $(\alpha^{s'})^s = \beta^{s'}$, where s' is a randomly distributed shared secret that effectively "blinds" the decryption of (α, β). The aforementioned 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. To achieve this, 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. Then, three Distributed Exponentiation (DEx) protocols are executed (two for computing $\alpha' = \alpha^{s'}$ and $\beta' = \beta^{s'}$ and the last one to check that $(\alpha')^s = \beta'$). Such protocol on input α outputs α^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 protocol of [23] for proving equality of discrete logs, described in Appendix A. The protocol satisfies special soundness and is honest–verifier zero–knowledge [19]. 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 Λ denote any subset of $t+1$ participants who successfully passed the zero knowledge proof and let $\lambda_{j,\Lambda}$ denote the appropriate Lagrange interpolation coefficients. The desired value can be obtained from the following identities:

$$\alpha^s \;=\; \prod_{j\in\Lambda} \omega_j^{\lambda_{j,\Lambda}}\,, \qquad \lambda_{j,\Lambda} = \prod_{l\in\Lambda\setminus\{j\}} \frac{l}{l-j}\,.$$

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 $x \in \{x_0, x_1\}$ and wants to show that an ElGamal encryption of x, say $(\alpha, \beta) = (g^r, y^r x)$, is indeed of this form without revealing the value of x. The prover's task amounts to showing that the following relation holds:

$$\log_g \alpha \;\in\; \{\log_y(\beta/x_0), \log_y(\beta/x_1)\}\,.$$

Building on [16], an efficient witness indistinguishable (honest–verifier zero–knowledge) proof of knowledge for the above relation was proposed in [19]. For completeness sake we review it as well as a non–interactive version of it in Appendix B. Henceforth, **Proof–Ballot**$_{\{x_0,x_1\}}(\alpha,\beta)$ denotes this (non–interactive) proof.

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 describe a MIX–network proposal due to Abe [1] which in addition to the aforementioned properties also satisfies: correctness, robustness, privacy, and universal verifiability. MIX–servers first run the DKG protocol and jointly generate a secret s and a public y. Initially, the bulletin board 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 $m_l \in G_q$ and $t_{0,l} \in_R \mathbb{Z}_q$. (To avoid the attack shown in [49] a proof of knowledge of $t_{0,l}$ must accompany $(G_{0,l}, M_{0,l})$.) The list of ElGamal ciphertexts is re-randomized and permuted by the cascade of MIX–servers. Server j chooses a random permutation π_j of S, picks $t_{j,l} \in_R \mathbb{Z}_q$ for each l, 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

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

Processing proceeds sequentially through all servers.

Lemma 2.1 ([1]) *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, referred to as **Protocol**–Π, 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. A non–interactive version of **Protocol**–Π can be derived through standard techniques. We henceforth denote this (non–interactive) version by **Proof**–Π. (See details in [1].)

Our MTS can be based on any re-encrypting MIX–network with the mentioned characteristics. Other alternatives will be discussed later on.

3 Membership Testing Scheme (MTS)

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. 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.

Basic MTS Protocol

Input

1. Public Input: System parameters, i.e., a k bit long prime p, a prime $q > n$ that divides $p-1$ and generators g and h of an order q multiplicative subgroup G_q of $\mathbb{Z}_p^*$ (elements g and h are uniformly and independently generated). 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 bulletin board area.
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 $G_{0,l} = 1$ and $M_{0,l} = h^{-l}$ for each $l \in S$.
2. Authority A_j chooses at random a permutation π_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 $l \in S$,

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

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 $(\alpha_i, \beta_i) = (g^{r_i}, y^{r_i} h^{v_i})$, and **Proof–Ballot**$_{\{h^0, h^1\}}(\alpha_i, \beta_i)$. Each authority checks the proof. If the check fails the ballot is declared incorrect.

Output Phase

1. Each authority computes $\alpha = \prod_i \alpha_i$ and $\beta = \prod_i \beta_i$ over all correctly issued ballots.

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 $(G'_l)^s = M'_l$ for some l in S. In the affirmative case they output MEMBER, otherwise NON–MEMBER.

Remark 3.1 *Note that both the MIX Phase and the Verification Phase may be pre-computed 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 $S = \{9, 10, 11, 12\}$.

4 Analysis

4.1 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.

4.2 No–duplication

Follows from requiring each voter to compute the challenge in the (non–interactive) proof of validity of ballots as a hash of, among others, a unique public key identifying the voter.

4.3 Correctness

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

Theorem 4.1 *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 Section 3, for $j \leq m$ let $\pi_{j,\ldots,m}$ denote $\pi_j \circ \pi_{j+1} \circ \ldots \circ \pi_m$ and π 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 $l = \pi^{-1}\left(\sum_i v_i\right)$, it holds that $\pi(l) = \sum_i v_i$. Hence, $h^{s'_l(\sum_i v_i - \pi(l))}$ equals 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. ∎

4.4 Robustness

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

Lemma 4.2 ([19]) *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 4.3 ([1]) **Protocol**–Π *is a honest verifier zero–knowledge proof of knowledge for π and $\tau_{m,l}$'s. The protocol is also honest verifier zero–knowledge proof of knowledge for π_j's and $t_{j,l}$'s held by honest provers.*

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

Theorem 4.4 *Assume there are at most $t < m/2$ 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. When an attempt by the adversary to force an incorrect output is detected, the identity of the authorities controlled by the adversary will be exposed with overwhelming probability.*

Proof: By Lemma 4.2 it suffices to consider the case where 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 4.3 insures that with overwhelming probability the Verification Phase will detect it, the tracing option invoked, and the identity of dishonest authorities exposed. ▮

4.5 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.[2]

Secure Function Evaluation Framework: We follow the model of secure multi-party function evaluation proposed by Canetti [13], and hence, we cast our Basic MTS protocol as computing a function f_S on the voters' votes, namely $f_S(v_1, \ldots, v_n) = b$, where b is the bit set to 1 if $\sum_i v_i \in S$ and 0 otherwise.

Very roughly, in [13] the notion of security of a cryptographic protocol $\mathcal{P}$ computing a function f is formalized by considering a setting with two different "worlds": the "real world" and the "ideal world". In the former, protocol $\mathcal{P}$ is executed on the presence of some adversary $\mathcal{A}$. In the latter, the following "ideal" protocol $\mathcal{I}$ is executed instead: each party (authority or voter) privately delivers her input to an (incorruptible) trusted party which then computes the desired function f on those inputs, and privately hands the corresponding output of the function back to each party. Protocol $\mathcal{P}$ is considered private and correct if, for every adversary $\mathcal{A}$ attacking protocol $\mathcal{P}$ in the real world, we can exhibit an adversary $\mathcal{SIM}$ (called the simulator) which can do "as well as $\mathcal{A}$" but *this time running on the ideal world*. In our setting, adversary $\mathcal{SIM}$ "doing as well as" $\mathcal{A}$ means that the combined output of adversary $\mathcal{A}$ and the honest parties after running $\mathcal{P}$ is computationally indistinguishable from the corresponding output of adversary $\mathcal{SIM}$ and honest parties after running the ideal protocol $\mathcal{I}$. This notion captures the property that protocol $\mathcal{P}$ cannot be less secure than the most secure protocol $\mathcal{I}$ computing the same function f. The reader is refereed to [13] for background and further details of the model.

In the analysis of this section we make the simplifying assumption that the signing key generation procedure for authorities and voters is executed as the very first step in the MTS protocol. This assumption is not essential and can be removed at the expense of a more involved argument.

The following two lemmas show that some of the building blocks of our Basic MTS protocol (see Section 2.2) do not leak more information than the function they compute.

Lemma 4.5 ([32, 41]) *Assume a broadcast channel is available and there are less than $t < m/2$ dishonest authorities. Then, under the Decisional Diffie Hellman assumption (DDH), the following hold:*

a. *The DKG protocol privately and correctly computes the function* DKG*, which is defined as "On any input, generate the authorities' shares $s_1, \ldots, s_m$ of a random secret s, and value*

[2] An extension of this argument may also yield an alternative proof of the robustness of our protocol.

$y = g^s$. *Each authority receives as output one share* s_i *and the value* y. *Each voter receives a copy of* y."

b. *The DEx protocol privately and correctly computes the function* Exp *defined as "On input the authorities' shares of a secret* s *and a value* G, *compute the value* G^s. *The output is given to all the parties."*

Moreover, the simulator implied by part (a) *(respectively by part* (b)*) satisfy the condition that, on input a randomly chosen input* y^*, *it outputs a distribution indistinguishable from an execution of DKG (respectively DEx) where the public output is* y^*.

Lemma 4.6 *Assume a broadcast channel is available and there are less than* $t < m$ *dishonest authorities. Then, under the Decisional Diffie Hellman assumption (DDH), the MIX–network protocol (ie.* MIX Phase *and* Verification Phase*) privately and correctly computes the function* MIX *defined as defined as: "On any input, compute a random permutation of the list* $(h^{-\ell})_{\ell \in S}$ *and then encrypt each element of the list. All parties receive the resulting list as output."*

Proof: It suffices to consider an ideal adversary $\mathcal{SIM}_{MIX}$ that simulates the work of all honest authorities while using the real adversary $\mathcal{A}$ to drive the strategy for the dishonest authorities. The view of adversary $\mathcal{A}$ in the simulation is indistinguishable from the one in a real execution of the protocol, otherwise we can either break the semantic security of ElGamal encryption (or equivalently, the DDH assumption) or contradict the zero-knowledge property of **Protocol**–Π. Similarly, since $t < m$, Lemma 4.3 and the semantic security of ElGamal encryption guarantee that the output of the honest authorities and voters in the simulation follow the same distribution as in the real execution of the MIX protocol. ▌

MTS PROTOCOL IS PRIVATE: We conclude this section by showing that protocol MTS does not leak any more information besides that implied by the output of the scheme. Specifically, the following holds

Theorem 4.7 *Assume there are less than* $t < m/2$ *dishonest authorities and* n' *dishonest voters controlled by an adversary* $\mathcal{A}$. *For any such adversary* $\mathcal{A}$ *attacking MTS, there exits a simulator* $\mathcal{SIM}$ *such that, for any list* $(v_i)_i$ *of votes cast by the voters, the following two distributions are computationally indistinguishable under the Decisional Diffie Hellman assumption:*

- *The output of adversary* $\mathcal{A}$ *running on the real world when protocol MTS is executed on input* $(v_i)_i$, *and*
- *The output of simulator* $\mathcal{SIM}$ *running on the ideal world where the trusted party computes function* f_S *over private inputs* $(v_i)_i$.

Proof: Let $\mathcal{SIM}_{DKG}$, and $\mathcal{SIM}_{DEx}$ denote the simulators for protocols DKG, and DEx given by Lemma 4.5. We build a simulator $\mathcal{SIM}$ for the ideal protocol computing function f_S. Simulator $\mathcal{SIM}$ has black-box access to adversary $\mathcal{A}$ and works as follows: First, it simulates the first step of the Setup Phase of the MTS protocol by running the signing key generation procedure on behalf of the honest authorities, picking a random value s and running $\mathcal{SIM}_{DKG}$ on target value $y = g^s$.

(Note that such simulation allows $\mathcal{SIM}$ to know the "shared secret" s.) Analogously, shared values s'_ℓ are computed by invoking $\mathcal{SIM}_{DKG}$ once for each ℓ.

Then, $\mathcal{SIM}$ runs $\mathcal{SIM}_{MIX}$ to simulate the MIX Phase and Verification Phase. $\mathcal{SIM}_{MIX}$ outputs a list $(G_{m,\ell}, M_{j,\ell})_\ell$.

Once in the Voting Phase, $\mathcal{SIM}$ chooses random votes for all honest voters and simulate their work during this phase (i.e., signing key generation, ballot preparation and ballot posting). By querying adversary $\mathcal{A}$, the simulator obtains the ballots for the corrupted parties. Using the knowledge of secret s, the simulator extracts the values v_i of the correctly cast votes. These values are used as input to function f_S which returns a bit b.[3]

Then, for each $\ell \in S$, $\mathcal{SIM}$ runs simulator twice $\mathcal{SIM}_{DEx}$; first on input $G_{m,\ell}$ and target value $G'_\ell = (G_{m,\ell})^{s'_\ell}$, and then on input $M_{m,\ell}$ and target value $M'_\ell = (M_{m,\ell})^{s'_\ell}$. If $b = 0$, simulator $\mathcal{SIM}$ picks a random value $\hat{M}$ in G_q (the simulation fails if $\hat{M} = M'_\ell$ for some $\ell \in S$); otherwise, if $b = 1$, simulator picks $\ell^* \in S$ at random and sets $\hat{M} = M'_{\ell^*}$. The simulator concludes by running $\mathcal{SIM}_{DEx}$ on input G'_ℓ and target value $\hat{M}$.

By Lemma 4.5 and Lemma 4.6, the simulation of the Setup, MIX and Verification phase is correct. For the Voting Phase, the semantic security of the ElGamal encryptions and the zero-knowledge property of the **Ballot–Proof** guarantee adversary $\mathcal{A}$ cannot distinguish a simulated view from a view in the real protocol. Finally, for the Output Phase we make the realistic assumption that $q >> N$. In such case, Lemma 4.5 guarantees adversary $\mathcal{SIM}$ can, without being detected, control the simulation so as to satisfy the condition $(G'_\ell)^s = M'_\ell$ for some $\ell \in S$ if and only if $b = 1$. ▮

4.6 Universal Verifiability

Follows from the public verifiability of the proofs of ballot validity (**Proof–Ballot**), the proof of randomization and permutation (**Proof**–Π), the proof of knowledge of equality of discrete logarithms (**Proof–Log**) and the correctness proof associated to the DKG protocol. Note that even in the case that there are more than t dishonest authorities, although privacy might be compromised, passive observers will still be able to ascertain whether the protocol was correctly performed.

4.7 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 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 communication 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

[3] Recall that in the ideal model of [13], even though the simulator is not allowed to access the honest parties' inputs, all such inputs are indeed used by the trusted party to compute function f_S.

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 [19].

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

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

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 (before the voting begins). The communication 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, more efficient MIX–networks like the one proposed by Desmedt and Kurosawa [25] might be used instead of Abe's MIX–network in the MTS of Section 3. In this case, the work done by each authority during the pre-computation stage is reduced to $O(kN)$. 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 more efficient recent MIX–network proposals like those of Neff [44], Furukawa and Sako [28], Abe [2, 3], and Jakobsson and Juels [37], are good candidates for drop in replacements in the MIX module of the MTS of Section 3. (All the latter preserve universal verifiability.)

After submission of this paper, MIX–network proposals were put forth that relax the universal verifiability property while significantly improving their overall efficiency [34]. Also, it has been shown that efficiency can be improved if a non–negligible probability of error in the verification process can be tolerated [40] (the probability of error goes down as the number of elements permuted increases, but still this is probably unacceptable in the jury voting scenario).

Other Homomorphic Encryption Schemes: Similarly, the requirements from the encryption scheme used in this work are being a (semantically secure) homomorphism between plaintexts and ciphertexts, and an efficiently identifiable encryption of 1. Thus, cryptosystems like the one proposed by Paillier [47] can also be used to instantiate our MTS. The MIX network of Furukawa and Sako [28], the proofs of equality of encryption and the proofs of ballot validity can be easily adapted in order to be implemented based on Paillier's system. The resulting scheme can be proven secure using an analysis similar to the one of Section 4.

UNANIMITY VOTING: In case of unanimity voting S is a singleton. Therefore, there is no need to use a MIX–network. Thus, the computational cost of the scheme is reduced by skipping the pre-computation phase.

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. 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.

PRIVATE INFORMATION RETRIEVAL: When restricted to a single voter V with a vote $v \in S' \supset S$, the proposed scheme yields a method of searching for v on the "database" S without revealing neither the target v nor the contents of the database. The corresponding proof of validity can be designed using 1-out-of-n proofs [16] as suggested in [18]. This problem is a special case of what is known as private information retrieval [15, 29]. It encompasses situations where users are likely to be highly motivated to hide what information they query from a database that contains particularly sensitive data, e.g., stock quotes, patents or medical data.

7 Final Comments

An interesting feature of our electronic jury voting scheme is that it combines parallel lines of research concerning electronic voting, one based on MIX–networks [21] and another on homomorphic encryptions [17, 7, 5]. 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 [17, 7, 5] which does not rely on MIX–networks.

Acknowledgments

We would like to thank Daniele Micciancio, Rosario Gennaro, Mihir Bellare, Meaw Namprempre, Bogdan Warinschi, and Tadayoshi Kohno for interesting and helpful discussions, Martin Loebl for suggesting the application to the scoring problem, an anonymous referee for bringing to our attention [35] and suggesting the PIR application, and all the reviewers for comprehensive comments that helped us improve the presentation of this work.

References

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

[2] M. Abe. Mix–Networks on Permutation Networks. *Proc. of Asiacrypt'99*, 258–273. Springer-Verlag. LNCS Vol. 1716.

[3] M. Abe, and F. Hoshino. Remarks on Mix–Networks on Permutation Networks. *Proc. of PKC'01*, 317–324. Springer-Verlag. LNCS Vol. 1992.

[4] O. Baudron, P. A. Fouque, D. Pointcheval, G. Poupard and J. Stern Practical Multi-Candidate Election System. *Proc. of CCS'01*, 274–283. ACM.

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

[6] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. *Proc. of STOC'94*, 544–553, ACM.

[7] J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. *Proc. of PODC'86*, 52–62. ACM.

[8] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. *Proc. of STOC'88*, 1–10. ACM.

[9] D. Boneh and P. Golle. Almost entirely correct mixing with applications to voting. *Proc. of CCS'02*, 68–77. ACM.

[10] F. Boudot, B. Schoenmakers and J. Traore. A fair and efficient solution to the socialist millionaires' problem. *Discrete Applied Mathematics*, 111(1-2):23–36, 2001.

[11] F. Boudot. Efficient proofs that a committed number lies in an interval. *Proc. of EuroCrypt'00*, 431–444. Springer-Verlag. LNCS Vol. 1807.

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

[13] R. Canneti. On the composition of multiparty protocols. *Journal of Cryptology*, 13(1): 143–202. Springer-Verlag.

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

[15] B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. Private information retrieval. *Proc. of FOCS'95*, 41-50. IEEE.

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

[17] J. D. Cohen and M. J. Fischer. A robust and verifiable cryptographically secure election scheme. *Proc. of FOCS'85*, 372–382, IEEE.

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

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

[20] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. *Proc. of EuroCrypt'99*, 90–106. Springer-Verlag. LNCS Vol. 1592.

[21] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981. ACM.

[22] D. Chaum. Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. *Proc. of EuroCrypt'88*, 177–182. Springer-Verlag. LNCS Vol. 330.

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

[24] I. Damgård and M. Jurik. A generalization, a simplification and some applications of Paillier's probabilistic public-key system. *Proc. of PKC'01*, 119–136. Springer-Verlag. LNCS Vol. 1992.

[25] Y. Desmedt and K. Kurosawa. How to break a practical MIX and design a new one. *Proc. of EuroCrypt'00*, 557–572. Springer-Verlag. LNCS Vol. 1807.

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

[27] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *AusCrypt'92*, 244–251, 1992. Springer-Verlag. LNCS Vol. 718.

[28] J. Furukawa and K. Sako. An efficient scheme for proving a shuffle. *Proc. of Crypto'01*, 368–387. Springer-Verlag. LNCS Vol. 2139.

[29] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *Proc. of STOC'98*, 151-160. ACM.

[30] R. Gennaro. Manuscript, 1995.

[31] R. Gennaro. Achieving independence efficiently and securely. *Proc. of PODC'95*, 130–136, ACM.

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

[33] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. *Proc. of FOCS'86* 174–187, IEEE.

[34] P. Golle, S. Zhong, D. Boneh, M. Jakobsson and A. Juels. Optimistic mixing for exit–polls. *Proc. of Asiacrypt'02*, 451–465. Springer-Verlag. LNCS Vol. 2501.

[35] M. Hirt and K. Sako. Efficient receipt–free voting based on homomorphic encryption. *Proc. of EuroCrypt'00*, 539–556. Springer-Verlag. LNCS Vol. 1807.

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

[37] M. Jakobsson and A. Juels. Millimix: Mixing in small batches. Technical Report 99–33, DIMACS, 1999.

[38] M. Jakobsson and A. Juels. Mix and match: Secure function evaluation via ciphertexts. *Proc. of Asiacrypt'00*, 162–178. Springer-Verlag. LNCS Vol. 1976.

[39] M. Jakobsson and A. Juels. An optimally robust hybrid mix network. In *PODC'01*, 284–292. ACM.

[40] M. Jakobsson, A. Juels and R. Rivest. Making MIX–nets robust for electronic voting by randomized partial checking. *Prof. of USENIX Security Symposium'02*, 339-353. USENIX.

[41] S. Jarecki. *Efficient Threshold Cryptosystems*. PhD thesis, MIT, LCS, June 2001.

[42] T. Malkin. *A Study of Secure Database Access and General Two-Party Computation*. PhD thesis, MIT, LCS, Feb. 2000.

[43] M. Mitomo and K. Kurosawa. Attack for flash MIX. *Proc. of Asiacrypt'00*, 192–204. Springer-Verlag. LNCS Vol. 1976.

[44] C. A. Neff. A verifiable secret shuffle and its application to e-voting. *Proc. of CCS'01*, 116–125. ACM..

[45] W. Ogata, K. Kurosawa, K. Sako and K. Takatani. Fault tolerant anonymous channel. *Proc. of ICICS'97*, 440–444. Springer-Verlag. LNCS Vol. 1334.

[46] M. Ohkubo and M. Abe. A length-invariant hybrid mix. *Proc. of Asiacrypt'00*, 178–191. Springer-Verlag. LNCS Vol. 1976.

[47] P. Paillier. Public–key cryptosystems based on composite degree residuosity classes. *Proc. of EuroCrypt'99*, 223–238. Springer-Verlag. LNCS Vol. 1592.

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

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

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

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

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

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

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

A Proof of Knowledge for Equality of Discrete Logarithms

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

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

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

B Proof of Validity of Ballot

Voter		Verifier
$r, w, \rho_{1-v}, d_{1-v} \in_R \mathbb{Z}_q$		
$a_{1-v} \leftarrow g^{\rho_{1-v}} \alpha^{d_{1-v}}$		
$b_{1-v} \leftarrow y^{\rho_{1-v}} (\beta / x_{1-v})^{d_{1-v}}$		
$a_v \leftarrow g^w$		
$b_v \leftarrow y^w$	$\xrightarrow{\alpha,\beta,a_0,b_0,a_1,b_1}$	
$d_v \leftarrow c - d_{1-v}$	$\xleftarrow{c}$	$c \in_R \mathbb{Z}_q$
$\rho_v \leftarrow w - r d_v$	$\xrightarrow{d_0,d_1,\rho_0,\rho_1}$	$c \stackrel{?}{=} d_0 + d_1$
		$a_v \stackrel{?}{=} g^{\rho_v} \alpha^{d_v}$, $\forall v \in \{0,1\}$.
		$b_v \stackrel{?}{=} y^{\rho_v} (\beta / x_v)^{d_v}$, $\forall v \in \{0,1\}$.

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

In order to make the protocol of Figure 2 non–interactive, the challenge c is computed as a hash of the values α, β, a_0, b_0, a_1, b_1. This maintains security in the random oracle model. In order to prevent vote duplication, the challenge must be made voter–specific. Following [31] this can be done by having voter V_i compute the challenge as the hash of the values ID_i, α, β, a_0, b_0, a_1, b_1, 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 [1], denoted **Protocol**–Π, 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–Π
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 commitment 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 $j = 0$, φ_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 π_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 (the resulting protocol remains secure under the random oracle model). In this case, the non–interactive proof consists of the outputs of the last server.