Distributing the Power of a Government to Enhance the Privacy of Voters
(Extended Abstract)

Josh Cohen Benaloh*
Yale University

Moti Yung†
Columbia University

1 Introduction

In this paper, we distribute the functions of the government in the cryptographic election scheme of [CoFi85]. In so doing, we are able to achieve privacy of individual votes in a much stronger sense without giving up any of the previously attained properties of robustness or verifiability. This gives an electronic mechanism for holding a large-scale election amongst untrusted parties which is far more useful in "real-world" applications than the previous scheme.

The combination of distributed computing and cryptography has become increasingly important. Applications of cryptography to distributed computing have been seen recently in [Bro85], [FeMi85], and [CGMA85]. Secret-ballot elections seem to be a natural problem in distributed computing. [DLM82], [Mer83], and [Yao82] proposed completely distributed boardroom election schemes in which participants pass encrypted messages around a room and then decrypt the messages to reveal the election tally. These schemes suffer from a lack of robustness in that if any of the participants fails or otherwise misbehaves during the election process, the scheme fails to produce any tally whatsoever.

Chaum in [Cha81] proposed a centralized scheme using a "mix" which scrambles messages to protect the identities of the senders. Chaum also comments about the possibility of cascading several mixes to enhance the security of his system. The idea of a mix provides an elegant solution to the election problem. However, no work has yet been done towards proving such a scheme secure with respect to some underlying set of cryptographic assumptions.

The centralized election scheme proposed in [CoFi85] achieves the desirable robustness and verifiability properties, and these properties are proven to hold with high probability. The privacy property, however, is not as strong as is required for a practical election because votes, while being secure from other voters, are easily readable by the central government.

By distributing the functionality of the central government of [CoFi85], the scheme presented here protects the privacy of individual votes. No one component of the government can determine how an individual voter voted. Furthermore, even if only one component of the government is honest, a conspiracy of all remaining government components is still not sufficient to compromise the privacy of individual votes. The accuracy of the ultimate tally can be verified by all participants, and no set of voters can prevent such a tally from being produced.

In addition, the scheme presented here achieves a higher confidence than the prior scheme. Previously, the chance of an incorrect tally being produced was inversely proportional to a security parameter which was polynomially related to the run time of the scheme. In this scheme, the chance of an incorrect tally is inversely proportional to an exponential function of the same security parameter. The previous scheme also made extensive use of a trusted "beacon" to generate public random bits. We show here how this function can be distributed among the agents already acting in the scheme. Finally, we present an alternate mechanism for carrying out multiway elections which is simpler than in the original work.

*Supported in part by the National Security Agency under Grant MDA904–84–H–0004.

†Supported in part by the National Science Foundation under Grant MCR-83-03139 and by an IBM graduate fellowship.

As has been the standard in cryptographic protocols, our security is dependent on a number-theoretic assumption. The privacy property in this scheme is reduced to a weak version of the so called *residue problem*. The reduction is proved rigorously, and the reduction is valid regardless of the actual complexity of the residue problem — a problem which has no known efficient solution. Other cryptographic applications have been based on versions of this number-theoretic problem. (See [Rab79], [GoMi84], [LMR83], [GMR85], and [GHY85] for just a few of the many examples.) The robustness and verifiability properties are proven independently of any cryptographic assumptions.

2 Model

We assume here that the agents of the protocol include a set of ω voters $V = \{v_1, v_2, \ldots, v_\omega\}$, a set of κ tellers (government components) $T = \{t_1, t_2, \ldots, t_\kappa\}$. Each of these agents is assumed to be a probabilistic polynomial-time process with added communication abilities.

The processes communicate by means of a broadcast network which can be regarded as a set of publically readable bulletin boards — one board owned by each process. Only the owner of a bulletin board can write to it, and no message can be erased from a board.

We also assume the existence of a global clock which is used to indicate a concluding time for each round of the protocol.

Extensive use is made of the existence of a generally trusted source of randomness. Such a source may be obtained either by the use of a "beacon" (see [Rab83]) or by a sub-protocol executed by some subset of the participants of the protocol.

The scheme depends upon the value of a security parameter N. By increasing N, the security of the scheme can be increased arbitrarily, but the running time of the scheme may also be dependent on N.

The scheme defines a Boolean predicate $\mathcal{G}$ on the space of all possible message sequences produced by a voter process. Informally, $\mathcal{G}$ describes whether or not the message sequence appears to be consistent with an actual run of the protocol.

Each voter protocol actually consists of two protocols: a *no-protocol* and a *yes-protocol*. A voter which follows a no-protocol is said to cast a *no-vote*. A voter which follows a yes-protocol is said to cast a *yes-vote*.

The actual *tally* (t) of an election is the number of processes on which $\mathcal{G}$ evaluates to "true" and which cast yes-votes in the election.

Some value (Γ) produced by the election scheme is denoted as the tally. An election scheme is said to be *correct* if the tally computed by the protocol (Γ) matches the actual tally (t).

Privacy in an election is not absolute, as in a unanimous election, for example, everyone knows how everyone voted. All that can be claimed is that an honest yes-vote and an honest no-vote cannot be distinguished. More precisely, an election scheme is ε-secure if there is no probabilistic polynomial-time procedure which if given a pair of honest voters in an election — one of whom cast a yes-vote and one of whom cast a no-vote — can tell which is the yes-vote with probability greater than $\frac{1}{2} + \varepsilon$.

Given this definition, it is easy to show that for an ε-secure election scheme, there is no probabilistic polynomial-time procedure which can, with probability greater than $\frac{1}{2}+\varepsilon$, distinguish between any two assignments of votes to voters among any subset of honest voters with the same sub-tally over that set.

3 A Distributed Election Scheme

We extend the election paradigm of [CoFi85] by distributing the government into a set of tellers. The paradigm has eight basic phases. In phase 1, each teller selects and posts a pair of election parameters. In phase 2, each voter selects and reveals a set of unmarked "test ballots" — each consisting of an encrypted yes-vote and an encrypted no-vote. In phase 3, each voter "marks" each test ballot by randomly selecting one of the two votes. In phase 4, the tellers decrypt all voters' test votes. In phase 5, each voter releases its own decryptions of its test votes. Any discrepancies between a voter's decryptions in phase 5 and a teller's decryptions in phase 4 indicate the invalidity of that teller and void the election. At this point, we may assume that if an election continues beyond phase 5, then with very high probability, all tellers' parameters are valid. In phase 6, each voter selects and reveals an unmarked "master ballot" again consisting of an encrypted yes-vote and an encrypted no-vote. In phase 7, each voter "marks" its master ballot by designating one of the two votes as desired. Finally, in phase 8, the tellers compute the tally of the master votes, and release information to prove that the tally is correct.

The key new idea which allows the government to be distributed comes from considering the government as holding a counter which each voter has the opportunity to increment or leave unchanged.

Such a counter may be distributed by having each of the κ tellers hold a single counter. The global interpretation is that the sum of values held by all of the local counters, when considered modulo a fixed prime, gives the value of the global counter. Votes are constrained so as to either increment the global counter by 1 (modulo the fixed prime) or to leave it unchanged.

To give very high confidence that the election parameters, ballots, and tally are of the required form, extensive use is made of interactive proofs. Interactive proofs are becoming a common tool for multiparty protocols. These "proofs" are themselves protocols which employ random selection to convince users of specified properties while protecting additional private information. For details and formal definitions of interactive proof methods, the reader is referred to [GMR85].

3.1 Definitions and Overview

Let r be prime. Fix n and let y be relatively prime to n such that y is not an r^{th}-residue modulo n (i.e. there exists no x such that $y \equiv x^r \pmod{n}$). Such an (n, y) pair is said to be a *valid* pair. If (n, y) is a valid pair, an integer z which is relatively prime to n is expressible in the form $y^e x^r$ for at most one e in the range $0 \leq e < r$. Let $\mathsf{type}[z, (n, y)]$ be defined on such z to be this unique value e (if defined). Define n to be *admissible* if $n = pq$ where p and q are prime, $r|(p-1)$, and $r \not| (q-1)$. If n is admissible and (n, y) is valid, then $\mathsf{type}[z, (n, y)]$ is defined for all z which are relatively prime to n. It is believed that deciding r^{th}-residuosity modulo n is hardest when n is admissible.

Let $P = \langle (n_1, y_1), (n_2, y_2), \ldots, (n_\kappa, y_\kappa) \rangle$ be a vector of pairs such that each y_i is relatively prime to n_i and is not an r^{th}-residue modulo n_i. Such a P is said to be a *valid* parameter set. Let $Z = \langle z_1, z_2, \ldots, z_\kappa \rangle$ be a vector of integers such that each z_i is relatively prime to n_i. We define

$$\mathsf{TYPE}[Z, P] = \left(\sum_{i=1}^{\kappa} \mathsf{type}[z_i, (n_i, y_i)] \right) \bmod r.$$

Such a vector Z is called a *vote*.

A vote Z is said to be *valid* (relative to a given P) if $\mathsf{TYPE}[Z, P] \in \{0, 1\}$. A vote Z of type 0 (relative to P) is called a *no-vote*, and a vote Z of type 1 (relative to P) is called a *yes-vote*, A pair of votes is called a *ballot*. An (i, j)-ballot is defined to be a ballot containing a vote of type i and a vote of type j. A ballot is said to be *valid* if it consists of one no-vote and one yes-vote — i.e. a $(0, 1)$-ballot.

Two integers z_1 and z_2 which are of the same type with respect to a valid (n, y) pair can be proven so by releasing an x such that $z_2 \equiv z_1 x^r \pmod{n}$. Two votes Z_1 and Z_2 can be shown to be of the same type with respect to P by showing that the vote formed as the componentwise quotient of Z_1 and Z_2 is of type 0. Two ballots B_1 and B_2 may be shown to be of the same (i, j)-type (*type-equivalent*) by showing that one of the votes in B_1 is of the same type as the first vote in B_2 and the other vote in B_1 is of the same type as the second vote of B_2. The technical details of proving that two ballots are type-equivalent without revealing undesirable information are actually somewhat subtle and will be given in section 3.4.

To conduct an election, each teller t_i begins by producing a valid (n_i, y_i) pair. A voter votes in an election by preparing a valid ballot, interactively proving its validity, and selecting one of the two votes as its actual vote. Each teller "collects" all vote components corresponding to its chosen parameters and computes the sum of their types. The sum (taken modulo r) of all of the tellers' "subtallies" represents the total number of yes-votes cast in the election.

In the following sections, we will describe in detail how such an election can be conducted. By adding appropriate restrictions and incorporating interactive proofs in various stages of the process, we will show how the validity of the election tally can be verified by all participants. We prove this result and also prove that under a number-theoretic assumption about the difficulty of deciding residuosity, the votes of voters remain private even under strong assumptions about possible conspiracies among agents.

If a trusted source of random bits is assumed, then the tally of an election is proved to be correct with very high probability, and no further assumptions are required. If, instead, participants follow a subprotocol to generate bits, then some sort of probabilistic one-way function is required. The assumption about the difficulty of deciding residuosity is sufficient for this purpose.

The following lemma is an immediate consequence of the properties of residue classes and the definition of the TYPE function. It shows that the sum of the types of two votes is given by the type of their componentwise product. Thus, the type of the componentwise product of all valid votes cast in an election will give the tally of the election.

Lemma 1 *If P is a valid parameter set and if $A = \langle a_1, a_2, \ldots, a_\kappa \rangle$ and $B = \langle b_1, b_2, \ldots, b_\kappa \rangle$ are votes of type α and β, respectively, then $\mathsf{TYPE}[A \cdot B, P] \equiv \alpha + \beta \pmod{r}$ where $A \cdot B$ is defined to be the vote*

given by the componentwise product of A and B with the i^{th} component take modulo n_i.

3.2 The Scheme

We divide the actual protocol into eight phases corresponding to the outline described above. Fix a prime number r such that r is greater than the number of potential voters. Let N be the security parameter of the system. Assume that a set T of tellers and a set V of eligible voters are fixed in advance. (Note that not all eligible voters need participate.)

Phase 1:
Each teller $t_i \in T$ randomly selects an admissible $n_i = p_i q_i$ with the additional constraint that each of p_i and q_i are N-bit primes. Each teller t_i then randomly selects a y_i such that $\gcd(n_i, y_i) = 1$ and y_i is not an r^{th}-residue modulo n_i.

Each teller then posts this (n_i, y_i) pair.

Phase 2:
Each voter randomly selects N valid "test" ballots of the form previously defined. The voter then demonstrates that each of these N ballots is indeed valid by engaging in N interactive proofs with the beacon.[1]

To interactively prove that a given "primary" test ballot is valid, a voter prepares N valid "auxiliary" ballots at random and a bit from the beacon is then associated with each. For those auxiliary ballots for which the beacon's bit is 0, the voter decrypts the ballot to demonstrate that it is valid. The voter then proves that all auxiliary ballots for which the beacon's bit is 1 are type-equivalent by proving that each is type-equivalent to the given primary ballot. By showing that all undecrypted auxiliary ballots are type-equivalent and all decrypted auxiliary ballots are valid, the voter demonstrates that, with very high probability, at least one (and therefore all) of the undecrypted auxiliary ballots are valid. Since these undecrypted auxiliary ballots have been shown to be type-equivalent to the given primary ballot, there is also very high confidence that it too is valid. The details of proving ballots type-equivalent are given in section 3.4.

Phase 3:
Each voter designates one vote from each of its test ballots. Whether to designate the no-vote or the yes-vote is decided randomly for each ballot.

Phase 4:
Each teller t_i decrypts and reveals the type of the i^{th} component of each of the test votes. This proves, with very high probability, the teller's ability to distinguish between vote-types and thereby demonstrates that the y_i selected by teller t_i is in fact not an r^{th}-residue modulo n_i, as required.

Phase 5:
Each voter reveals the decryption of all of its test votes. If any voter reveals a vote decryption which differs from a teller's decryption (i.e. a teller asserts that a vote component is of type e_1 and a voter later decrypts it showing it to be of type $e_2 \not\equiv e_1 \pmod r$), then that teller's parameters may be assumed invalid, and the election is voided.

The election is continued beyond phase 5 *only* if no discrepancies among the test votes are revealed.

Phase 6:
Each voter randomly selects a valid "master" ballot of the form previously defined. The voter then demonstrates that this master ballot is valid by engaging in an interactive proof as in phase 2.

Phase 7:
Each voter designates one vote from its master ballot. To vote "no", the voter designates the no-vote. To vote "yes", the voter designates the yes-vote.

Phase 8:
Each teller compiles the portions of votes corresponding to its parameters as follows:

Teller t_i computes $W_i \equiv \prod z_{(i,j)} \pmod{n_i}$, where $z_{(i,j)}$ is the i^{th} component of voter v_j's designated vote from its master ballot. This product, however, is taken over only those votes cast by voters which have satisfied the consistency predicate $\mathcal{G}$. Teller t_i then releases a τ_i such that W_i is expressible as

$$W_i \equiv y_i^{\tau_i} X_i \pmod{n_i}$$

where X_i is an r^{th}-residue modulo n_i. Finally teller t_i engages in an interactive proof to show that X_i is in fact an r^{th}-residue. This shows that $\mathsf{type}[W_i, (n_i, y_i)] = \tau_i$. The method for proving interactively that an integer is an r^{th}-residue modulo a given n is shown in section 3.3.

This completes the protocol.

The tally of an election is

$$\Gamma \equiv \sum_{i=1}^{\kappa} \tau_i \pmod r.$$

[1] We will show in section 3.5 how the beacon can be simulated by a sub-protocol executed by the tellers.

The consistency predicate $\mathcal{G}$ is defined to be true for voter v_j if and only if both of the following conditions hold:

> In phase 6 of the election, voter v_j posts a master ballots together with N auxiliary ballots as prescribed, successfully demonstrates that each auxiliary ballot which was required to be opened is valid, and also demonstrates that each remaining auxiliary ballot is type-equivalent to the master ballot.
>
> In phase 7 of the election, voter v_j designates one vote from its master ballot.

Note that we do not require that a voter partake in phases 2, 3, or 5 to satisfy $\mathcal{G}$. These phases are designed to test the validity of the tellers' parameters, and the consistency predicate $\mathcal{G}$ is only required to ensure that, with very high probability, the actual master vote cast by a voter is valid.

3.3 Proving that a Given Integer is an r^{th}-residue

In order to prove that a given integer z is an r^{th}-residue modulo n, an agent which possesses an x such that $z \equiv x^r \pmod{n}$ could, by releasing such an x, easily demonstrate that z is an r^{th}-residue. Releasing this r^{th}-root of z could, however, give an adversary additional information which might aid in factoring n. Instead, the agent can engage in the following sub-protocol using a beacon to facilitate an interactive proof.

The agent begins by randomly selecting N integers c_i, $1 \le i \le N$, such that all c_i are relatively prime to n. The agent then forms $C_i \equiv c_i^r \pmod{n}$, for all i and releases the C_i. Next a beacon bit b_i is associated with each C_i. For each i such that $b_i = 0$, the agent reveals c_i. For each i such that $b_i = 1$, the agent reveals $c_i x$.

By releasing c_i such that $C_i \equiv c_i^r \pmod{n}$ for each i such that $b_i = 0$, the agent gives observers very high confidence that at least one of the remaining C_i is also an r^{th}-residue. When C_i is an r^{th}-residue relatively prime to n, z is an r^{th}-residue if and only if $C_i z$ is an r^{th}-residue. By releasing $c_i x$, the agent releases an r^{th}-root of $C_i z$, thereby demonstrating that $C_i z$, and therefore z, is an r^{th}-residue. Since the $C_i z$ formed in this way represent random r^{th}-residues, releasing random roots of random residues gives an adversary no information which it could not compute by itself. The formal proof that this interactive sub-protocol does not release undue information is subsumed by the proof of privacy given in section 5.

Proving that an integer z is an r^{th}-residue modulo n demonstrates that $\mathsf{type}[z,(n,y)] = 0$ for any given y which is not an r^{th}-residue modulo n. To show that $\mathsf{type}[z,(n,y)] = e$ for some e, $0 \le e < r$, it suffices to show that $\mathsf{type}[zy^{-e},(n,y)] = 0$. Thus, an agent which possesses an x such that $z \equiv y^e x^r \pmod{n}$ can, by proving interactively that zy^{-e} is an r^{th}-residue, demonstrate that z is of type e (with respect to n and y).

3.4 Proving Type-Equivalence of Ballots

Recall that a vote Z is a vector $\langle z_1, z_2, \ldots, z_\kappa \rangle$ of integers such that each z_i is relatively prime to n_i. Two votes $A = \langle a_1, a_2, \ldots, a_\kappa \rangle$ and $B = \langle b_1, b_2, \ldots, b_\kappa \rangle$ are type-equivalent if and only if (when considered modulo r)

$$\sum_{i=1}^{\kappa} (\mathsf{type}[a_i,(n_i,y_i)] - \mathsf{type}[b_i,(n_i,y_i)]) \equiv 0.$$

This equivalence can be demonstrated by revealing the difference e_i (taken modulo r) between the type of each a_i and of each b_i and showing that $\sum_{i=1}^{\kappa} e_i \equiv 0 \pmod{r}$. Each e_i is, in turn, equivalent to the type (modulo r) of $a_i/b_i \pmod{r}$. (If $a \equiv y^{e_a} x_a^r$ and $b \equiv y^{e_b} x_b^r$, then $a/b \equiv y^{e_a - e_b} (x_a/x_b)^r \pmod{n}$.)

Section 3.3 describes how a given integer z can be shown to be of a given type e when given an x such that $z \equiv y^e x^r$. This case is somewhat simpler, however, since there is no harm in releasing the known root x. Thus, a given a_i/b_i is shown to be of type e by releasing an x such that $a_i/b_i \equiv y^e x^r \pmod{n}$, but there is a subtlety here.

The mechanism by which a voter generates the requisite a_i and b_i gives the voter x_a, x_b, e_a, and e_b such that $a_i \equiv y^{e_a} x_a^r \pmod{n}$ and $b_i \equiv y^{e_b} x_b^r \pmod{n}$. Thus,

$$\frac{a_i}{b_i} \equiv y^{e_a - e_b} \left(\frac{x_a}{x_b}\right)^r \pmod{n},$$

and x_a/x_b suffices as a root which can be released. However, if $\mathsf{type}[a_i,(n_i,y_i)] < \mathsf{type}[b_i,(n_i,y_i)]$, then the associated $e_a - e_b$, the type of their quotient (before being normalized modulo r), is less than 0. It would not be wise to reveal this information, since it *does* give an adversary information about the relative types of a_i and b_i and could thereby give an adversary an estimate of the type of a_i.

To avoid this problem, when $\mathsf{type}[a_i,(n_i,y_i)] < \mathsf{type}[b_i,(n_i,y_i)]$, we observe that

$$\frac{a_i}{b_i} \equiv y^{e_a-e_b}\left(\frac{x_a}{x_b}\right)^r \equiv y^{e_a-e_b}y^r\left(\frac{x_a}{x_b y}\right)^r \pmod{n}.$$

Thus, it is still possible to reveal an e with $0 \le e < r$ such that $(a_i/b_i)y^{-e}$ can be demonstrated to be an r^{th}-residue modulo n by using $x = x_a/(x_b y)$.

The type of a_i/b_i is released (and proven) for all i such that $1 \le i \le \kappa$. If the sum of these types is 0 modulo r, A and B are type-equivalent.

Recall, finally, that a ballot is a pair of votes. To show that two ballots are type-equivalent then, it is necessary only to show that the first vote of the first ballot is type-equivalent to one of the votes of the second ballot and the second vote of the first ballot is type-equivalent to the other vote of the second ballot. The use of ballots here and the means by which they are shown to be equivalent actually describe a special case of cryptographic capsules. For a discussion of cryptographic capsules, see [Coh86b].

3.5 The Beacon and its Simulation

Various aspects of the election scheme (including the sub-protocols of sections 3.3 and 3.4) made extensive use of a generally trusted source of random bits. Such bits are required both in phases 2 and 6 of the election (in order to verify the validity of ballots) and in phase 8 (allowing the tellers to prove that their sub-tallies are of the claimed type).

Rabin in [Rab83] introduces the notion of a beacon. A beacon is a physical device which generates and broadcasts bits (either at fixed intervals or upon demand) which are truly random and which can be read by all participants to a protocol. The use of such a device makes the model and election scheme simpler to understand; however, the beacon is not strictly necessary to this scheme.

Instead of using a beacon, the tellers can execute a sub-protocol which effectively simulates a beacon and generates random bits. Recall that the tellers (as well as the voters) are probabilistic polynomial-time processes and therefore have private sources of random bits. We assume, as already required to ensure privacy of votes, that at least one teller is honest.

Each time a random bit b is required, each teller t_i generates a random bit b_i and a random x_i relatively prime to its n_i. If $b_i = 0$, teller t_i releases a $z_i = x_i^r$, otherwise, t_i releases $z_i = y_i x_i^r$.

After all tellers have posted their z_i's, each teller releases its x_i. From this, everyone can compute all b_i's. The desired random bit b is then the XOR of the b_i's.

Under the assumption about weak residue problem, b is a random bit if *no* y_i is an r^{th}-residue modulo n_i and at least one teller is honest. If at least one voter is honest, then a teller t_i whose y_i *is* an r^{th}-residue modulo n_i will be, with very high probability, exposed as dishonest before the conclusion of the protocol, regardless of the random bits selected by this protocol. In addition, if the tellers challenge each other with test ballots (as did the voters), then even the assumption of an honest voter becomes unnecessary.

The use of such a sub-protocol, however, makes the correctness of the tally (as well as the privacy of the votes) dependent on the cryptographic assumption about the difficulty of computing r^{th}-residues.

3.6 Multiway Elections

Although the discussion thus far has been limited to elections with only two possible choices, elections between multiple choices are also possible with very minor modifications to the scheme. The basic idea is to provide one counter for each choice.

Let C be the number of choices presented to the voters in an election. A valid ballot now consists of C votes, one of which is of type 1 and the remaining $C-1$ of which are of type 0. Once the validity of its ballot has been verified, a voter votes by assigning one vote from its ballot to each counter. The counter to which the vote of type 1 is assigned corresponds to the choice which the voter designates.

The resulting tally left in a given counter represents the total number of votes cast in favor of the corresponding choice.

4 Correctness

It is easy to see that all phases of the election protocol can be completed in time bounded by a small polynomial in N — the security parameter — and with a constant number of communication rounds. It must be shown that if the tellers follow their prescribed protocols, then the tally computed will represent the number of yes-votes cast in the election.

The following theorem depends on the existence of a trusted source of random bits. If a beacon is to be simulated, as in section 3.5, then the simulated beacon bits are random if at least one teller and at least one voter are honest and if there exist no probabilistic polynomial-time algorithm to decide residuosity.

Theorem 1 *If all tellers follow their protocols and at least one voter is honest, then, with very high probability, the tally* Γ *produced by the protocol is equal to the actual tally of the election. If the tally is not of this form, then with very high probability it will be detected.*

Proof:

First, if some teller t_i releases an (n_i, y_i) pair such that y_i is not an r^{th}-residue modulo n_i, then the probability that a given honest voter will detect a discrepancy on one of its test ballots is $1 - 2^{-N}$. There is, by assumption, at least one honest voter, and since more than one dishonest teller merely increases the likelihood of exposure, we may conclude that in an election which has not been voided in phase 5, the probability that all y_i's are not r^{th}-residues modulo their respective n_i's is at least $1 - 2^{-N}$.

Recall next that the actual tally of an election (as defined in section 2) is the number of processes on which $\mathcal{G}$ evaluates to "true" and which cast yes-votes in the election.

In phase 6, each voter completes an interactive proof that its master ballot is valid — contains one no-vote and one yes-vote. If all tellers' parameters are valid, then a voter who attempts to prepare an invalid master ballot will be detected and excluded from the election with probability at least $1 - 2^{-N}$. Thus, a voter for which the $\mathcal{G}$ predicate answers true has, with probability at least $1 - 2^{-N}$, cast a valid vote. The probability that valid votes have been cast by *all* voters for which the $\mathcal{G}$ predicate is true is therefore at least $1 - \omega 2^{-N}$.

Each teller t_i then computes the product W_i of the i^{th} component of each vote cast by each voter for which $\mathcal{G}$ evaluates to true. By lemma 1, the type of this product has the unique type which is the sum (modulo r) of the types of the components. The interactive proof used here gives very high confidence (at least $1 - 2^{-N}$) that the claimed type of W_i is in fact its actual type. We denote this type by τ_i. Thus, given that all tellers' parameters are valid and that all votes cast are valid, the probability that all tellers release τ_i's which correctly represent the types of their respective W_i's is at least $1 - \kappa 2^{-N}$.

If some teller fails to behave as described in the preceding paragraph, the teller is presumed to be faulty, and the tally (if produced at all) cannot be expected to be correct.

$\Gamma = \sum \tau_i \pmod r$ now represents, with probability at least

$$(1-2^{-N})(1-\omega 2^{-N})(1-\kappa 2^{-N}) \geq 1-(1+\omega+\kappa)2^{-N},$$

the type (with respect to the selected parameter set P) of the vector $W =< W_1, W_2, \ldots, W_\kappa >$. $\mathsf{TYPE}[W, P]$ is in turn the sum of the types of the votes cast in the election by voters on which $\mathcal{G}$ evaluated to true. This is therefore the number of yes-votes cast. ∎

5 Privacy

Finally, we must show that the privacy of votes remains secure even in the face of possible conspiracy by other voters and tellers. The only assumption we make here is that at least one teller is honest and does not take part in such a conspiracy.

The following theorem gives a reduction showing that an algorithm which can in general compromise voter privacy in an election can be used to develop an algorithm which can solve a slightly weakened version of the residuosity problem — a problem for which no efficient solution is known. This problem amounts to the problem of distinguishing between numbers which are r^{th}-residues and numbers which are not r^{th}-residues modulo an admissible n. A precise formulation of the weak r^{th}-residue assumption is given in [CoFi85].

The proof described here is almost identical to that in [CoFi85]. By assuming the existence of one honest teller, we can view that teller as a central government and use the prior proof almost unchanged. The only change which is required (other than that just described) emanates from slight changes in this version of the scheme (phases 1 through 5) and from the use of tellers to replace the beacon.

The appendix describes an alternate proof approach which, although no easier than the proof given here, is of interest in its own right and develops machinery which may be of use elsewhere.

Theorem 2
If there exists a probabilistic polynomial-time algorithm which, for all admissible n*, can distinguish with probability at least* $\frac{1}{2} + \varepsilon_n$ *between some two honest voters' votes — one of which is a no-vote and one of which is a yes-vote — in elections in which at least one teller is honest and selects* n *as its modulus, then there exists a probabilistic polynomial-time algorithm which can, for all admissible* n*, correctly decide* r^{th}*-residuosity modulo* n *with probability at least* $\frac{1}{2} + \varepsilon_n$*.*

Proof: (sketch)

An algorithm which can distinguish between votes can be regarded as a procedure $\mathcal{P}$ which controls

some number of voter processes (excepting the two voters whose votes are in question) and as many as all but one of the teller processes (by assumption at least one teller is honest). This procedure "participates" in elections and outputs one of two values indicating its guess as to which of the two selected voters cast a yes-vote. Under the assumption that such a procedure $\mathcal{P}$ exists, we show how to simulate elections for $\mathcal{P}$'s benefit and make use of its output (or lack of output) to gain an advantage in deciding residuosity.

Now, suppose we are given an (n, z) pair, where n is admissible, and we wish to determine whether or not z is an r^{th}-residue modulo n. Suppose further that $\mathcal{P}$ has a probability of at least $\frac{1}{2} + \varepsilon_n$ of correctly distinguishing a yes-vote from a no-vote in an election in which one of the tellers it does *not* control has selected n as its modulus.

We run, together with $\mathcal{P}$, a "pseudo-election" by allowing each teller and voter not controlled by $\mathcal{P}$ to run its normal protocol with the following exceptions.

- One teller not controled by $\mathcal{P}$ is to use as its election parameters (n, z') where n is the given modulus and $z' \equiv z^i x^r \pmod{n}$ for randomly chosen chosen integers i and x such that $0 < i < r$ and x is relatively prime to n. Assume, without loss of generality, that this is teller t_1.
- Of the two voters in question, exactly one is to follow a yes-protocol and the other is to follow a no-protocol with the choice of which is to do each decided randomly.

The computation of z' described above is designed to ensure that z' is an r^{th}-residue modulo n if and only if z is and that z' is uniformly chosen among the residues (resp. non-residues) relatively prime to n. (This is where n is required to be admissible.)

The only difficulty we encounter in simulating such an election is that we do not have the factorization of n and therefore cannot (directly) convince the voters controlled by $\mathcal{P}$ that we know the types of the first components of their votes. Such knowledge is required by t_1 to decrypt test votes in phase 4 and to compute its sub-tally τ_1 in phase 8.

To overcome this problem, we observe that $\mathcal{P}$, as a probabilistic algorithm, may be regarded as a deterministic algorithm with an external random source. By fixing this source, we may run $\mathcal{P}$ to a certain point, freeze $\mathcal{P}$, and then complete this run of $\mathcal{P}$ with two or more different continuations of the protocol. In this manner, we may see what $\mathcal{P}$ does in response to several different inputs at the same time.

Thus, by changing the random bit selected by t_1 at various stages, we may attempt to change various global bits required in the interactive proofs of phases 2 and 6. If ever a voter answers any auxiliary ballot both by decrypting it (as required if the global bit is 0) and (on a different continuation of $\mathcal{P}$) by proving it type-equivalent to its associated primary ballot (as required if the global bit is 1), then the associated primary ballot is easily decrypted.

There are now two cases to be considered. First, if we succeed in decrypting all primary ballots of voters controlled by $\mathcal{P}$, then we may complete the simulation and observe the output of $\mathcal{P}$.

By a symmetry argument, if the election simulation is completed and if z' (and hence z) were a residue there would be no way for $\mathcal{P}$ to distinguish a yes-vote from a no-vote with probability greater than $\frac{1}{2}$ (since, in this case, every vote can be decrypted as *both* a yes-vote and a no-vote). Of course, $\mathcal{P}$ may be able to recognize that this is the case, but it is impossible for $\mathcal{P}$ to determine which voter was assigned the yes-vote, and therefore $\mathcal{P}$'s behavior will be independent of this assignment.

By assumption, on completed elections with parameter n, $\mathcal{P}$ correctly determines which of the two designated voters cast a yes-vote with an overall probability of at least $\frac{1}{2} + \varepsilon_n$.

Thus, $\mathcal{P}$'s output behavior is independent of the chosen assignment when z' (and hence z) is a residue, and $\mathcal{P}$'s output behavior varies (by at least ε_n) as the chosen assignment varies when z' (and hence z) is not a residue. Thus, by a statistical sampling of $\mathcal{P}$'s output on known values, we may gain an ε_n advantage in determining whether or not z is a residue modulo n.

If the second case, there is some primary ballot which we cannot decrypt. We begin by observing that if some teller releases an invalid parameter in phase 1, then, with very high probability, it will be detected in phase 5, and the election will not continue beyond this stage. Since up to this point, honest voters have not in any way indicated their vote preferences, $\mathcal{P}$ cannot gain more than an inverse exponential advantage at distinguishing votes unless all parameters are valid. Such an inverse exponential advantage at deciding residuosity is easily obtainable without $\mathcal{P}$'s help.

We may therefore assume that all election parameters are valid, and there are (again) two cases to be considered here: either we are unable to (by freezing and then continuing $\mathcal{P}$ with different bit choices by t_1) change at least one global bit in some interactive proof, or we are able to change at least one global bit in every interactive proof but (although the voter

completed the relevant interactive proof on the first run of $\mathcal{P}$) we are unable to get the voter to complete such an interactive proof on any successive runs. In the former case, it may be assumed with extremely high probability that $\mathcal{P}$ was able to read t_1's bit (before its decryption) and adjust its bits so as to force the global bit to remain constant. By another symmetry argument, this is only possible if z' (and hence z) is a non-residue. The latter case can be shown to be of low probability unless, once again, $\mathcal{P}$ was able to read t_1's encrypted bit. Hence, if we are not able to dercypt all primary ballots and complete the simulation, we may conclude that with very high probability z is not an r^{th}-residue. ∎

Corollary 1
If there exists a probabilistic polynomial-time algorithm which, for all admissible n, can distinguish with an ε_n advantage between some two assignments of votes to voters within a group of honest voters (with the same sub-tally over this group of voters) in elections in which at least one teller is honest and selects n as its modulus, then there exists a probabilistic polynomial-time algorithm which can, for all admissible n, decide r^{th}-residuosity modulo n with an ε_n advantage.

Proof:
Let H be a set of honest voters and let $h = |H|$. Let s and s' be two distinct assignment of votes (yes or no) to voters in H such that both s and s' have the same number of yes-votes. Clearly, there exists a sequence of "swaps" $s = s_0, s_1, s_2, \ldots, s_{k-1}, s_k = s'$ with $k < h (< r)$ such that, for all i, assignments s_i and s_{i+1} differ only in one pair of votes.

An algorithm which distinguishes between assignments s and s' with an ε_n advantage must, for some i, distinguish between assignments s_i and s_{i+1} with at least an ε_n/r advantage. Since r is constant, the central limit theorem implies that a fixed number of repeated samplings can boost this advantage to ε_n. The two assignments s_i and s_{i+1} now satisfy the conditions of theorem 2 and thereby yield the corollary. ∎

6 Discussion and Conclusion

By distributing the capabilities of the centralized government election protocol of [CoFi85], we have substantially increased the privacy of electors. In the previous work, the government was able to determine how every voter cast its vote in the election. In the new scheme, even if only one teller is honest, the votes of honest voters remain private.

One disadvantage of this scheme is that if a teller, at any point of the protocol, ceases execution or otherwise deviates from the protocol in a detectable way, then no tally is produced, and the election must be restarted from the beginning (without the corrupted teller). [Coh86a] presents a fault-tolerant version of this scheme in which the election can be continued and a tally produced even in the presence of faulty tellers.

[Coh86a] also describes a variant of the original centralized election scheme in which the government can announce the winner of an election and prove the result to the election participants *without* revealing the actual tally of the election. How to do this in a decentralized manner without allowing any agent to determine the actual tally remains open.

Distributing the functions of the government has both enhanced the privacy of voters and allowed the beacon of the previous work to be replaced by a distributed protocol which is more practically realizable. We feel that the technique of distributing the functionality of agents in the context of cryptographic protocols is of significant value.

Acknowledgements

We would like to express many thanks to Mike Fischer, who was a partner in the orignal work on which this extension is based and who offered help in many aspects of this work, and to David Wittenberg, who proofread this work and made many helpful suggestions and comments.

References

[Bro85] **Broder, A.** "A Provably Secure Polynomial Approximation Scheme for the Distributed Lottery Problem." *Proc.* 4th *ACM Symp. on Principles of Distributed Computing*, Minaki, ON (Aug. 1985), 136–148.

[Cha81] **Chaum, D.** "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms." *Comm. ACM 24*, 2, (Feb. 1981), 84–88.

[CGMA85] **Chor, B.**, **Goldwasser, S.**, **Micali, S.**, and **Awerbuch, B.** "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 383–395.

[CoFi85] **Cohen, J.** and **Fischer, M.** "A Robust and Verifiable Cryptographically Secure Election Scheme." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 372–382.

[Coh86a] **Cohen, J.** "Improving Privacy in Cryptographic Elections." *TR-454, Yale University, Department of Computer Science*, New Haven, CT (Feb. 1986).

[Coh86b] **Cohen, J.** "Cryptographic Capsules: A Disjunctive Primitive for Interactive Protocols." *TR-471, Yale University, Department of Computer Science*, New Haven, CT (Apr. 1986).

[DLM82] **DeMillo, R.**, **Lynch, N.**, and **Merritt, M.** "Cryptographic Protocols." *Proc.* 14th *ACM Symp. on Theory of Computing*, San Francisco, CA (May 1982), 383–400.

[FeMi85] **Feldman, P.** and **Micali, S.** "Byzantine Agreement in Constant Expected Time (and Trusting No One)." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 267–276.

[GHY85] **Galil, Z.**, **Haber, S.**, and **Yung, M.** "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 360–371.

[GoMi84] **Goldwasser, S.** and **Micali, S.** "Probabilistic Encryption." *J. Comput. System Sci. 28*, (1984), 270–299.

[GMR85] **Goldwasser, S.**, **Micali, S.**, and **Rackoff C.** "The Knowledge Complexity of Interactive Proof-Systems." *Proc.* 17th *ACM Symp. on Theory of Computing*, Providence, RI (May 1985), 291–304.

[LMR83] **Luby, M.**, **Micali, S.**, and **Rackoff, C.** "How to Simultaneously Exchange a Secret Bit by Flipping a Symmetrically-Biased Coin." *Proc.* 24th *IEEE Symp. on Foundations of Computer Science*, Tucson, AZ (Nov. 1983), 11–21.

[Mer83] **Merritt, M.** "Cryptographic Protocols." Ph.D. Thesis presented at *Georgia Institute of Technology* (Feb. 1983).

[Rab79] **Rabin, M.** "Digitalized Signatures and Public-key Functions as Intractable as Factorization." *MIT/LCS/TR-212, MIT Laboratory for Computer Science*, Cambridge, MA (Jan. 1979).

[Rab83] **Rabin, M.** "Transaction Protection by Beacons." *J. Comp. Sys. Sci. 27*, 2 (Oct. 1983), 256–267.

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

Appendix: An r-ary Probabilistic Public-Key Cryptosystem

Goldwasser and Micali, in [GoMi84], define a (binary) *probabilistic public-key cryptosystem* (PPKC) which offers many advantages over "traditional" public-key cryptosystems. Their PPKCs are built upon their notion of an *unapproximable trapdoor predicate* (UTP) which is a formal version of a cryptographic assumption such as the residuosity assumption used here in elections.

PPKCs encrypt messages one bit at a time by producing many bits for each bit to be encrypted. Strong properties are proven about messages encrypted as the concatenation of many encrypted bits. Roughly, these properties imply that for any PPKC, any function of a message that can be evaluated by an adversary after seeing the encryption of the message could have been evaluated (almost as successfully) by the adversary without even being given the encrypted message (i.e. the encryption of a message gives an adversary almost no *a posteriori* information about the message that it didn't have *a priori*).

Goldwasser and Micali propose *quadratic* residuosity (Given n and z, does there exist an x such that $z \equiv x^2 \pmod{n}$?) as the basis for a UTP and build a PPKC on this predicate.

The type function used in this election scheme may be used as the core of an *r-ary probabilistic public-key cryptosystem*. Instead of encrypting only one bit at a time, one of up to r distinct values may be encoded with a single encryption. This offers greater efficiency with (seemingly) comparible security.

The definition of a UTP generalizes easily to an *unapproximable trapdoor function* (UTF), and this allows a straightforward generalization of the Goldwasser and Micali binary PPKC into an r-ary PPKC. The security theorems also generalize directly.

It is not hard to see that the problem of evaluating the type of an integer (with respect to a given (n, y) pair) with probability greater than $\frac{1}{r} + \varepsilon$ is equivalent to the problem of distinguishing between elements chosen from one of two fixed types with probability greater than $\frac{1}{2} + \varepsilon$. Thus, it is reasonable to propose using the type function as a UTF given the underlying r^{th}-residuosity assumption (as used for elections).

The r-ary PPKC defined by using the type function as a UTF is a direct generalization of the binary PPKC proposed in [GoMi84] with an r-ary rather than a binary alphabet. The advantage is that, given a security parameter N, a k-bit message can be encrypted using $(N + \log_2 r)k/\log_2 r$ bits rather than Nk bits with (under the assumptions) comparable security.

Once this mechanism has been developed, an alternate proof of Theorem 2 is suggested. By concatenating the vote components received by each teller, this set of votes can be regarded as an encrypted message in an r-ary PPKC. The sub-tally released by each teller, known votes cast by voters controlled by procedure $\mathcal{P}$, and constraints on vote components given by knowledge of other components held by tellers controlled by $\mathcal{P}$ all serve as partial information about the message space of all possible votes. The security theorems of [GoMi84] (when suitably generalized to the r-ary case) show that such partial information is of no significant use in extracting further information not directly implied by the partial information (which of two vote assignments is actually the case is an example), unless the underlying assumption about residuosity is violated.

The election scheme presented here is, however, more complex than an r-ary PPKC since the interactive proofs of phases 2 and 6 (constraining the ballots to valid types) require additional interactions. It must be shown that such interactions give no additional knowledge to an adversary. [GMR85] and [GHY85] focus on these very issues. By showing that an election can be simulated for an adversary (as was done in the original proof of Theorem 2), one shows that an adversary gains no such additional knowledge and that therefore, the privacy of the voters is maintained.