Yale University
Department of Computer Science

A Robust and Verifiable Cryptographically Secure Election Scheme (Extended Abstract)

Josh D. Cohen Michael J. Fischer

YALEU/DCS/TR-416
July, 1985

A Robust and Verifiable Cryptographically Secure Election Scheme[†]

(Extended Abstract)

Josh D. Cohen Michael J. Fischer

Yale University
New Haven, Connecticut

1 Introduction

Cryptographic techniques are becoming important for a wide variety of distributed computing tasks where the processing agents are either unreliable or untrustworthy. (Cf. [DiHe76], [DLM82], [Rab83], [RSA78], etc.)

This paper describes a cryptographic scheme for holding a secure secret ballot election in which all communication is public. Voters cast their votes electronically, suitably encrypted, and a "government" releases a tally and a proof of its correctness which can be verified by all.

The scheme has several novel aspects. First, it is robust in the sense that no conspiracy of dishonest voters can prevent, with more than very low probability, the successful completion of the election. Second, the government, even acting in collusion with a conspiracy of dishonest voters, cannot release a false tally without being detected by every honest voter, except with very low probability. Third, if any conspiracy of dishonest voters can compromise privacy by gaining more than a very slight amount of information about how the honest voters voted, beyond that which is contained in the tally, then we can find an efficient algorithm for a certain number-theoretic problem which has no known efficient solution. The proof is by a polynomial-time reduction of the latter problem to the former, and the reduction is valid regardless of the actual complexity of the number-theoretic problem. Moreover, none of these properties depend on any unsubstantiated cryptographic assumptions, although compromising privacy is of course only as hard as the number-theoretic problem.

Our scheme introduces a new form of "interactive proof" by which one participant gives *passive* observers high confidence that certain claims are true without releasing related private information. Previous uses of interactive proof have required that both parties *actively* participate in the proof [GM83], [GMR85].

While we make intuitive appeal to notions such as "election", "voter", "cheating", "privacy", etc. in motivating our work, the actual theorems apply to a formal model in which the necessary concepts are precisely defined. Thus, the above claim about the difficulty of compromising the privacy of votes applies to our formally defined "conspiracy", which is a collection of processes in the model with certain well-defined powers and limitations. How closely our formal model corresponds to life in the real world is a question that cannot be treated mathematically, but by formalizing the problem, we can at least bring into focus the assumptions upon which our results depend.

Our election scheme consists of a set of processes called "voters" and a distinguished process called the "government". Processes communicate via publicly readable bulletin boards, one for each sender. All parties have access to a global clock, and every message is automatically time-stamped by the current value of the global clock when it is posted. Thus, every party to the scheme can see every message and can determine for each who posted it and when.[1]

In addition, our model includes a "beacon" (see [Rab83]) which is a publicly-readable source of random bits. For simplicity, we assume that the beacon operates only on demand. A process wanting a beacon bit posts a special request on its bulletin board. Within a bounded time thereafter, the beacon responds by placing a random bit, time-stamped and identified as coming from the beacon, on the sender's bulletin board.

In greater detail, our scheme satisfies the following properties:

1. Each voter has a function to check the election results. With very high probability, either

[†]This work was supported in part by the National Security Agency under Grant MDA904-84-H-0004.

[1]Implementing the automatic time-stamping feature can be difficult in practice. All we actually require is that all honest participants in the election be able to agree for each message whether or not it was posted prior to a given deadline, but even this presents practical difficulties.

the election passes the check and the announced tally of votes is correct, or the election fails the check and the government is not following its prescribed protocol. This holds even though some of the voters may be dishonest, where a dishonest voter is defined to be any voter process running some polynomial-time probabilistic algorithm other than the one prescribed by the protocol. Thus, with very high probability, no set of dishonest voters can invalidate an election or cast more than its share of votes.

2. If a set of conspiring voters (not including the government) can obtain more than a small epsilon advantage at compromising the privacy of the honest voters, then a probabilistic polynomial time solution exists to a certain number-theoretic problem that is believed to be hard.

We do *not* claim to protect the privacy of individual votes from the government, making our scheme unsuitable in many practical situations. However, the role of the government could be played by a black box which has its output carefully monitored and which is destroyed (along with its ability to compromise privacy) as soon as its essential tasks are completed. Insuring privacy from the government while retaining the desirable features of our scheme appears to be very difficult.

The problem to which privacy is reduced is a weak version of the so called *residue problem*. A number z is (by definition) an r^{th} residue modulo n if and only if there exists x such that $z \equiv x^r \pmod{n}$. Given numbers n and z with $z < n$, the question of whether or not z is an r^{th} residue modulo n is decidable (in fact all roots can be be found) with $\mathcal{O}(r(\log r)^2(\log\log r)(\log n))$ expected arithmetic operations in the ring of integers modulo n if the factorization of n is known. (See [Ben81], [Rab80].) Adleman and McDonnell in [AdMc82] (see also [APR83], [Adl80]) show that an oracle which takes an r and a z and determines whether or not z is an r^{th} residue (modulo n) can be used to generate an efficient (although not quite polynomial time) algorithm to factor n. Note that when $r = 2$, this problem is the problem of determining quadratic residues used by Rabin, Goldwasser and Micali, and others as the basis of provably secure cryptosystems and pseudo-random number generators ([GM82], [GMT82], [Rab79]).

In the weak r^{th} residue problem used here, r remains fixed but large (greater than the number of potential voters), and the algorithm is only required to work with high probability for integers of a restricted form (which we call "admissible"), and then only for an inverse polynomial sized fraction of such numbers. Nevertheless, no such probabilistic algorithm is known, and we believe this version of the problem is also hard.

1.1 Related Work

Various cryptographic schemes have been proposed for boardroom voting in which participants pass encrypted messages from one to another while performing encryption and decryption operations until a certain point is reached at which all are confident of the outcome of the vote ([DLM82], [Mer83], [Yao82]). These all share the problems that the active participants must be known in advance and if one participant stops following its protocol during the election, the election cannot be continued. (In our scheme, only the *potential* participants need be known in advance.)

Chaum [Cha81] proposes the use of a trusted "mix" (similar to our "government") to scramble pairs of votes and digital pseudonyms. The votes are publicly revealed, but the identity of the corresponding voters is protected by the mix. Our scheme has properties very similar to those of a mix, but the approach is very different. Instead of hiding voters, the scheme in this paper hides the actual values of the votes. There is far less interaction with the government than with the mix paradigm, and, perhaps most importantly, we are able to prove our claimed privacy properties.

2 Definitions

We sketch below the model we have in mind for the election problem and the conditions that we want a solution to have.

2.1 Verifiable Elections

A *J-voter election system* $\mathcal{E}$ is an asynchronous system of $J + 1$ communicating processes together with a *beacon* and a *global clock*. The processes may be thought of as probabilistic Turing machines extended with operations for communication. The program run by such a process is called a *protocol*. One process G is designated as the *government*, and the other process $v_1, \ldots, v_J$ are designated as (potential) *voters*. We denote the set of voter processes by V. G and V are fixed in advance, and each process knows the designation of every process. The beacon is a secure and trusted source of independent unbiased random bits. The global clock provides a publicly readable nondecreasing function of real time t that is bounded above and below by polynomials in t. Processes are asynchronous, but we assume the step times are polynomially related to real time. Thus, there exist polynomials

p, q such that each process takes at least $p(t)$ and at most $q(t)$ steps up to time t.

Communication is via *bulletin boards* which can be thought of as restricted shared memories. Each process controls one bulletin board, which it is said to *own*. The correspondence between bulletin boards and processes is fixed in advance and known to all processes. Each bulletin board can be read by every process, but it can only be written by its owner and by the beacon, and then only by appending new messages, not by altering old ones. Messages are automatically time-stamped with the current value of the global clock and are posted to the bulletin board in time-stamp order. The beacon only writes in response to a special "beacon-request" message which the owner can post. Within a bounded time after such a request, the beacon posts a random bit to the requester's bulletin board, suitably time-stamped and signed by the beacon.

In addition, there is a publicly-readable number N called a *security parameter* which serves as the (only) input to each process. N controls the liklihood that the election is correct and that privacy is maintained.

A *J-voter election scheme* $\mathcal{S}$ consists of a collection of protocols for use by a J-voter election system, a polls-closing time function t_{close}, a termination time function t_{end}, and a function *check*. t_{close} and t_{end} depend only on N and satisfy $0 < t_{\text{close}}(N) < t_{\text{end}}(N)$. The function *check* returns either good or bad and depends on N and the messages posted to the public bulletin boards before time $t_{\text{end}}(N)$. *check* must be computable in time polynomial in N.

$\mathcal{S}$ prescribes a protocol π_G for the government process and two possible protocols for each voter: π_{yes} to be used to cast a "yes" vote and protocol π_{no} to be used to cast a "no" vote.

An *election* under $\mathcal{S}$ consists of a run of a J-voter election system $\mathcal{E}$ for which *check* returns good. Any process of $\mathcal{E}$ which follows (one of) its protocol(s) prescribed by $\mathcal{S}$ is said to be *proper*; otherwise it is *improper*. We say that a voter *casts a valid "yes" vote* (resp. *"no" vote*) if the messages it posts up to time $t_{\text{close}}(N)$ are consistent with the protocol π_{yes} (resp. π_{no}). We say it *votes properly* if it casts a valid "yes" or "no" vote; otherwise it *votes improperly*. Note that a proper voter by definition always votes properly, but an improper voter may or may not vote properly, and if it votes improperly, that fact may or may not be detectable by others.

The *tally* of an election is the pair $(t_{\text{yes}}, t_{\text{no}})$ where t_{yes} and t_{no} are the number of voters who cast valid "yes" and "no" votes, respectively. As part of its protocol, the government releases a pair $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$ which is said to be *correct* if $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}}) = (t_{\text{yes}}, t_{\text{no}})$.

Let δ be a function of N. The scheme $\mathcal{S}$ is said to be *verifiable with confidence* δ if, for any election system $\mathcal{E}$, *check* satisfies the following properties for random runs of $\mathcal{E}$ using security parameter N:

1. If the government is proper in $\mathcal{E}$, then, with probability at least $1 - \delta(N)$, *check* returns good and the government releases a correct pair $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$.
2. Whether or not the government is proper, the joint probability is at most $\delta(N)$ that *check* returns good and the government releases an incorrect pair $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$ (or fails to release any pair at all).

$\mathcal{S}$ is said to be *verifiable* if it is verifiable with confidence δ for some inverse polynomial function[2] δ.

2.2 Public Voting

A simple example of a verifiable election scheme is one in which each voter publicly posts a single "yes" or "no" vote by a certain deadline $t_{\text{close}}(N)$, chosen large enough to insure that each proper voter has sufficient time to post its vote. Thus, a vote is valid if it is the only message on its bulletin board that was posted before time $t_{\text{close}}(N)$. The government then counts the valid "yes" and "no" votes and announces the totals. $t_{\text{end}}(N)$ is chosen large enough to give the government sufficient time to examine all of the voters' bulletin boards and to compute and post the tally.

check returns good if and only if the totals of the valid votes are the same as those announced by the government. Thus, by computing the function *check*, any participant can verify the accuracy of the announced tally.

2.3 Privacy

Preserving privacy in an election does not always imply the inability of one voter to determine another's vote. For example, in the case of a unanimous mandate, every voter knows every other voter's vote. More generally, any coalition of voters can determine the subtally of the remaining voters by subtracting their own votes from the totals. We say that privacy is maintained if any conspiracy of voters has at most a small advantage at distinguishing between any two vote assignments that have the same subtally on the set of proper voters.

In order to account for the possibility of a collection of improper voters acting in concert, we augment our

[2] δ is said to be an *inverse polynomial function* if $\delta(N) = 1/p(N)$ for some non-constant polynomial p with positive leading coefficient.

model to permit private communication channels between improper processes. However, we do not want to assume that private channels are always available, so we do not permit their use in the protocols prescribed by a verifiable election scheme. In other words, the adversaries can communicate secretly among themselves but the proper processes cannot.

To formalize the privacy requirement, let $C \subseteq V$ be a set of voter processes (the conspirators), and let $c_0 \in C$. We define a (c_0, C)-*consipracy* $\mathcal{C}$ to consist of an assignment of protocols to processes in C (possibly with private communication among themselves) such that c_0 produces an output in $\{0,1\}$ which we denote "output($\mathcal{C}$)". We require the running time of each process in C to be polynomial in N.

We now define what it means for a conspiracy $\mathcal{C}$ to compromise privacy of a J-voter election scheme $\mathcal{S}$. Let $H = V - C$ (the honest voters), and let h_0 and h_1 be assignments of votes to voters in H such that the subtally of votes in h_0 is the same as that in h_1. For each $i \in \{0,1\}$, define an election system $\mathcal{E}_i$ as follows. Every process in C runs the protocol assigned to it by $\mathcal{C}$, every voter process $v_j \in H$ runs protocol $\pi_{h_i(j)}$, and the government process G runs its proper protocol π_G. For a fixed security parameter N, let p_i be the probability that output($\mathcal{C}$) $= 1$ on a random run of $\mathcal{E}_i$, and let ε be a real number. We say that $\mathcal{C}$ *distinguishes* h_0 *from* h_1 *with* ε *advantage* if

$$|p_1 - p_0| > \varepsilon.$$

In other words, the conspiracy has an ε advantage in determining whether the votes correspond to assignment h_0 or to h_1 in a given election with security parameter N.

We say that $\mathcal{C}$ *compromises the privacy of* (h_0, h_1) *in* $\mathcal{S}$ if, for some inverse polynomial function ε, $\mathcal{C}$ distinguishes h_0 from h_1 with $\varepsilon(N)$ advantage on infinitely many values of N. Finally, we say that $\mathcal{S}$ is *secure* if for every (c_0, C)-conspiracy $\mathcal{C}$ and every pair of vote assignments h_0, h_1 to voters in $V - C$ that have the same subtally, $\mathcal{C}$ does not compromise the privacy of (h_0, h_1) in $\mathcal{S}$.

The *election problem* is to find, for each J, a J-voter election scheme that is verifiable and secure.

3 A Provably Secure Election Scheme

We now describe our election scheme. It is based on a simple paradigm, which we present first. Then we describe our means of encrypting votes, and finally we give the complete protocol.

3.1 An Election Paradigm

The basic election paradigm upon which our scheme is based operates in four phases and is shown in Figure 1. The participants are the government G and a set of voters $V = \{v_1, v_2, \ldots, v_J\}$. Implicit in the paradigm is that each phase must be completed by all participants before the next begins. This is achieved by setting deadlines in advance for the completion of each phase.

1.	G:	Select and reveal a set of parameter values S to be used in the election and interactively prove that S conforms to certain specifications.
2.	Each v_j:	Select and reveal an unmarked ballot B_j consisting of one encrypted "yes" vote and one encrypted "no" vote in random order and interactively prove that B_j is of this form.
3.	Each v_j:	Select one vote as the actual vote on the ballot B_j.
4.	G:	Release the tally and an interactive proof that the tally is correct.

Figure 1: The election paradigm.

3.2 Encryption of Votes

The basic idea behind the election scheme that we present in the next section is the encoding of votes by certain integers modulo a number n of a restricted form.

Fix r throughout to be a prime number greater than J, the number of potential voters. Let N as usual be the security parameter. We say n is N-*admissible* if it is the product of two primes p and q of length N for which $r|(p-1)$ and $r\not|(q-1)$. Let n be N-admissible and let y be relatively prime to n. We say w is an i-*vote* (with respect to r, n, and y) if w is relatively prime to n, $0 \le i < r$, and $w \equiv y^i x^r \pmod{n}$ for some integer x, and we say w is a *vote* if it is an i-vote for some i. The following lemma gives conditions under which w is an i-vote for a unique i in the range $0 \le i < r$.

Lemma 1 *Let n be N-admissible. Let w and y be relatively prime to n, and assume that y is not an r^{th} residue modulo n. Then w is an i-vote for exactly one integer i in the range $0 \le i < r$.*

An i-vote is said to be *valid* if $i \in \{0,1\}$. 0-votes

(numbers of the form x^r) encode "no" votes, and 1-votes (numbers of the form yx^r) encode "yes" votes.

An (unmarked) *ballot* is a (randomly ordered) pair of votes. It is of *type* (i, j) if it consists of one i-vote and one j-vote. A ballot of type $(0, 1)$ is said to be *valid*. A set of ballots is *valid* if every member is valid.

There are two things of interest to note about unmarked ballots. First, a valid ballot $B = \{w, w'\}$ contains no information as to the intended vote of the voter, so an unused ballot can be proved valid simply by revealing two numbers f and g such that $B = \{f^r, yg^r\}$. Second, two ballots $\{v, v'\}$ and $\{w, w'\}$ can be proven to be of the same type either by exhibiting r^{th} roots modulo n of v/w and v'/w' or by exhibiting r^{th} roots modulo n of v/w' and v'/w. Thus, given a partition of a set X of valid ballots into two subsets Y and Z, one can prove that all ballots in Y are valid and all ballots in Z are of the same type (i, j), and this can be done without revealing any information as to whether any particular vote on a ballot in Z is an i-vote or a j-vote. This is the basis of an interactive proof of validity for a set of ballots. Since there is at most one partition of an invalid set of ballots into two subsets Y and Z such that Y is valid and all ballots in Z are of the same type, the chance that the proof will succeed on an invalid set of ballots for a randomly chosen partition is only $2^{-\eta}$, where $\eta = |X|$.

The ability of each voter to verify the tally depends on a structural property of i-votes. Lemma 1 defines a mapping $h : w \mapsto i$ that takes a vote w to the unique i in the range $0 \le i < r$ for which w is an i-vote. $h : Z_n^* \to Z_r$ is a homomorphism since if w_1 is an i_1-vote and w_2 is an i_2-vote, then $w_1 \cdot w_2$ is an $(i_1 + i_2)$-vote. The following lemma is immediate.

Lemma 2 *If each of $w_1, w_2, \ldots, w_k$ is a valid vote and $k < r$, then $W = \prod w_j$ is a t-vote, where*

$$t = |\{j : w_j \text{ is a "yes" vote}\}|.$$

Thus, to verify that $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$ is, with high probability, the correct tally of the election, it suffices to verify that, with high probability, the conditions needed by Lemma 1 hold, that the total number of valid votes is $\hat{t}_{\text{yes}} + \hat{t}_{\text{no}}$, and that the product of all valid votes is a $\hat{t}_{\text{yes}}$-vote. This is accomplished by incorporating appropriate "interactive proofs" into the protocol in a way that can be checked by the procedure *check*.

3.3 The Scheme $\mathcal{S}_0$

The complete scheme $\mathcal{S}_0$, based on the paradigm of Figure 1, is shown in Figure 2. η_1, η_2, and η_4 are parameters that affect both the chance that a bad election will go undetected and also the degree of privacy attained. In order to achieve the desired levels of confidence and privacy, we choose these parameters as integer functions of N and r as follows:

$$\begin{aligned} \eta_1(N, r) &= N, \\ \eta_2(N, r) &= \lceil \log_2(rN) \rceil, \\ \eta_4(N, r) &= \lceil \log_2(N) \rceil. \end{aligned}$$

The time allowed for each phase grows as N^6. Thus, we let $t_{\text{close}} = cN^6$ and $t_{\text{end}} = 2cN^6$ for appropriately chosen c. (The actual constant of proportionally of course depends on the assumed step time of the processors.)

In step 4a, $\mathcal{S}_0$ uses the function $check_V(j)$ which checks voter v_j. It returns good if and only if all of the following hold.

- In step 2a, voter v_j posts $\eta_2 + 1$ ballots $B_{j,i}$, $0 \le i \le \eta_2$, consisting of pairs of integers relatively prime to n. (We denote the first and second components of these pairs by $B_{j,i}[1]$ and $B_{j,i}[2]$, respectively.)
- In step 2c, for each i such that $b_i = 1$, voter v_j posts integers f_i and g_i such that $\{f_i^r, yg_i^r\} \equiv B_{j,i} \pmod n$, and for each i such that $b_i = 0$, v_j posts integers x_i and $\hat{x}_i$ such that either

  $$\{x_i^r,\, \hat{x}_i^r\} \equiv \{B_{j,i}[1]/B_{j,0}[1],\, B_{j,i}[2]/B_{j,0}[2]\}$$

  or

  $$\{x_i^r,\, \hat{x}_i^r\} \equiv \{B_{j,i}[1]/B_{j,0}[2],\, B_{j,i}[2]/B_{j,0}[1]\}$$

  as sets modulo n.
- In phase 3, the selected vote $w_j \in B_{j,0}$.

To complete the specification of our election scheme, we describe the procedure *check*. It returns good if and only if all of the following hold.

- In step 1a, the government releases η_1 pairs of integers (n_i, y_i) such that $1 < y_i < n_i$ and $2N - 1 \le |n_i| \le 2N$.
- In step 1c, for each $i \ne m$, the government releases primes p_i and q_i of length N whose product is n_i such that $r_i | (p_i - 1)$, $r_i \not| (q_i - 1)$, $\gcd(y_i, n_i) = 1$, and y_i is not an r^{th} residue modulo n_i.
- In step 4a, the government releases η_4 numbers C_i.

Phase 1 steps executed by government:

1a. Release η_1 randomly-chosen pairs (n_i, y_i) such that n_i is N-admissible, $\gcd(y_i, n_i) = 1$, and there exists no x_i such that $x_i^r \equiv y_i \pmod{n_i}$.

1b. Use beacon to generate a random integer m, $1 \le m \le \eta_1$.

1c. For all $i \ne m$, reveal length N primes p_i and q_i such that $n_i = p_i q_i$, $r_i | (p_i - 1)$, and $r_i \not| (q_i - 1)$. Denote (n_m, y_m) by (n, y).

Phase 2 steps executed by each voter $v_j \in V$:

2a. For $0 \le i \le \eta_2$, randomly select f_i and g_i such that $\gcd(f_i, n) = \gcd(g_i, n) = 1$, and release a ballot $B_{j,i}$ consisting of the two numbers $(f_i^r \bmod n)$ and $(y g_i^r \bmod n)$ in random order.

2b. Use beacon to generate η_2 random bits $b_i, 1 \le i \le \eta_2$.

2c. For all i such that $b_i = 1$, reveal f_i and g_i. For all i such that $b_i = 0$, reveal $f_i \cdot f_0^{-1}$ and $g_i \cdot g_0^{-1} \bmod n$.

Phase 3 executed by each voter $v_j \in V$:

3. Select one element of $B_{j,0}$ as the actual vote w_j. To vote "yes", select $w_j = y g_0^r$. To vote "no", select $w_j = f_0^r$.

Phase 4 steps executed by government:

4a. Compute $\Gamma = \{j : \mathit{check}_V(j) = \mathsf{good}\}$ and $W = \prod_{j \in \Gamma} w_j \bmod n$. Randomly select η_4 numbers c_i such that $\gcd(c_i, n) = 1$ and reveal all $C_i = c_i^r$.

4b. Use beacon to generate η_4 random bits $b_i, 1 \le i \le \eta_4$.

4c. Compute x and t such that $W \equiv y^t x^r \pmod{n}$ and $0 \le t < r$. Reveal $(t, |\Gamma| - t)$. For all i such that $b_i = 1$, reveal c_i. For all i such that $b_i = 0$, reveal $c_i x$.

Figure 2: The election scheme $\mathcal{S}_0$.

- In step 4c, the government releases a pair (t, t') such that $t + t' = |\Gamma|$, where $\Gamma = \{j : \mathit{check}_V(j) = \mathsf{good}\}$; for each i such that $b_i = 1$, the government releases c_i such that $c_i^r \equiv C_i$; and for each i such that $b_i = 0$, the government releases an integer c_i' such that $y^t (c_i')^r \equiv C_i \prod_{j \in \Gamma} w_j \pmod{n}$.

4 Correctness

In this section, we sketch briefly the proof that the scheme $\mathcal{S}_0$ presented in Section 3 is a verifiable J-voter election scheme as defined in Section 2.

4.1 Termination within the Allowed Time Bounds

The protocol comprises a number of "primitive operations" on length N numbers. A *primitive operation* is a basic arithmetic operation modulo n, a greatest common divisor computation, or the computation of an r^{th} root modulo $n = pq$, where p and q are known. The basic arithmetic operations and gcd can easily be performed in time $\mathcal{O}(N^3)$. Each r^{th} root can be found using $\mathcal{O}(r(\log r)^2(\log\log r)(\log n))$ expected modulo n arithmetic operations as mentioned in Section 1. Since r is fixed independent of N and n is of length $\mathcal{O}(N)$, the total expected time is $\mathcal{O}(N^4)$.

All of the steps of $\mathcal{S}_0$ except for 1a require at most $\mathcal{O}(N)$ primitive operations; hence the total expected time for all of these steps is $\mathcal{O}(N^5)$.

In step 1a, values of p, q, and y must be chosen of the required form. The expected number of y's to be tested is a constant since only 1 out of every r of the possible y such that $\gcd(y, n) = 1$ are r^{th} residues. A generalization of the prime number theorem guarantees that both among integers p such that $r | (p - 1)$ and among integers q such that $r \not| (q - 1)$, the density of primes is about $\frac{1}{N}$ (see [Kra84, page 23]). Since primality testing (with high confidence) can be completed in $\mathcal{O}(N^3)$ time, a p and q of the desired form can be found in $\mathcal{O}(N^4)$ expected time. Thus, $\eta_1 = N$ such pairs can be found in expected time $\mathcal{O}(N^5)$.

We see that the total expected time required by both the government and the voter protocols is $\mathcal{O}(N^5)$. It follows that by proper choice of c, the fraction of runs in which some proper process fails to complete its protocol in time cN^6 can be made smaller than $1/N$. Hence, the voter protocols finish by time t_{close} and the government protocol finishes by time t_{end} with probability at least $1 - 1/N$.

4.2 Correctness of the Tally

The main tool in showing that the announced tally is correct is the use of interactive proofs in steps 1, 2, and 4. The basic idea of these interactive proofs is to force an agent who would produce fraudulent information to "outguess" the beacon, which can be done successfully with only low probability.

In phase 1, the government prepares η_1 pairs of parameters of which only one will be used. All of the remaining pairs must be of the required form or the government will be detected as being improper. Unless the government can successfully guess which pair will be used, it cannot subvert an election by planting parameters which do not conform to the specifications of the scheme. By making η_1 sufficiently large, we can make the probability of this occurring arbitrarily small.

In phases 2 and 4, a more sophisticated interactive proof following the ideas of [FMR84] and [GMR85] is used in which the probability of successful cheating is exponentially small. In these cases, undetected sabotage by a single voter would require predicting the values of all η_2 beacon bits, which of course can happen with probability only $2^{-\eta_2}$, so the probability that at least one of a set of k conspirators can successfully cheat is at most $k2^{-\eta_2} \leq r2^{-\eta_2}$.

The interactive proofs thus give high confidence $(1-1/\eta_1 = 1-1/N)$ that the parameter values chosen by the government satisfy the specifications and high confidence $(1 - r2^{-\eta_2} \geq 1 - 1/N)$ that no vote is improper. It must still be shown that the government cannot release a false tally.

Since, by lemma 2, the product of valid votes yields a t-vote, where t is the correct tally of the "yes" votes in the election, the released tally must be correct unless one or more of the interactive proofs admits a fraudulent value. This is used to derive the following lemma.

Lemma 3

1. *With probability at least* $(1-1/(rN))$, $check_V(j) =$ good *if and only if voter* v_j *votes properly.*
2. *If the government is proper in* $\mathcal{E}$*, then, with probability at least* $1 - 2/N$*, check returns* good *and the government releases a correct pair* $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$*.*
3. *Whether or not the government is proper, the joint probability is at most* $3/N$ *that check returns* good *and the government releases an incorrect pair* $(\hat{t}_{\text{yes}}, \hat{t}_{\text{no}})$ *(or fails to release any pair at all).*

Proof: (sketch)

1. If v_j votes properly, then $check_V(j) =$ good. If v_j does not vote properly, then $check_V(j) =$ bad unless the interactive proof of phase 2 succeeds, which occurs with probability at most $2^{-\eta_2} \leq 1/(rN)$.

2. Assume the government is proper. By the timing analysis in Section 4.1, it fails to complete its protocol within the allowed time bound with probability at most $1/N$. If it does complete its protocol, the released tally is correct unless the wrong votes are counted. By part 1 of this lemma, the probability that the vote of a voter who votes improperly is counted or the vote of a voter who votes properly is not counted is at most $1/(rN)$, so the probability that some voter is miscounted is at most r times that, or $1/N$. Hence, with probability at least $1 - 2/N$, the government completes its protocol, counts the correct votes, and releases a correct tally. In such a case, *check* always returns good.

3. The government can release an incorrect tally and still have *check* return good only if either the parameters n, y used to conduct the election do not meet the required specifications, or the wrong votes are counted, or the government succeeds in phase 4 in "proving" that $\prod w_j \bmod n$ is a t-vote for some t other than the correct value. The probability that bad parameters will be used and not detected is at most $1/\eta_1 = 1/N$. From part 1, the probability that the wrong votes are counted is at most $r2^{-\eta_2} \leq 1/N$. Finally, the probability of a false proof in phase 4 going undetected is at most $2^{-\eta_4} \leq 1/N$. Thus, the chance that $check =$ good and the released tally is incorrect is at most $3/N$. ∎

The following is an immediate consequence of Lemma 3.

Theorem 1 $\mathcal{S}_0$ *is verifiable with confidence* $3/N$*.*

5 Security

It remains to show that if the votes of proper voters can be compromised by a conspiracy of improper voters, then our weak r^{th} residue problem (defined below) can be efficiently solved. Recall that we are assuming here that the government is proper, for we have no protection against an improper government reading a vote and divulging it to others.

The proof is by a reduction. Assuming a conspiracy $\mathcal{C}$ exists that can compromise the election scheme $\mathcal{S}_0$, then we construct a probabilistic polynomial time algorithm to solve the weak residue problem.

5.1 The Weak r^{th} Residue Problem

Let $\mathcal{B}$ be a probabilistic expected polynomial time algorithm that takes two inputs n and z with z less than n and produces a single bit of output. We say $\mathcal{B}$ is a *weak*

r^{th} *residue tester* if there exists an inverse polynomial function ε' such that for infinitely many numbers N, for an $\varepsilon'(N)$ fraction of the N-admissible numbers n, and for all $z < n$,

$$\text{Prob}[\text{output}(\mathcal{B}) = 1 \Leftrightarrow z \text{ is an } r^{\text{th}} \text{ residue}] > 1 - \varepsilon'(N).$$

In other words, $\mathcal{B}$ can distinguish with very high probability between r^{th} residues and non-r^{th} residues for a significant fraction of the admissible numbers of infinitely many different lengths.

5.2 The Reduction

Let $\mathcal{C}$ be a (c_0, C)-conspiracy that compromises the privacy of (h_0, h_1) in $\mathcal{S}_0$. We construct a probabilistic polynomial time algorithm $\mathcal{B}$ that solves the weak r^{th} residue problem.

Recall from Section 2 the construction of the systems $\mathcal{E}_i$, $i \in \{0, 1\}$, from the conspiracy $\mathcal{C}$ and the vote assignments h_0 and h_1. Let $p_i(n)$ be the probability that $\mathcal{C}$ outputs 1 on runs of $\mathcal{E}_i$, given that the first parameter chosen in phase 1 is n. Since $\mathcal{C}$ compromises the privacy of (h_0, h_1) in $\mathcal{S}_0$, there are infinitely many values of N for which $\mathcal{C}$ distinguishes h_0 from h_1 with an ε advantage, where ε is an inverse polynomial function.

Consider such an N. We have by definition that

$$|p_0 - p_1| > \varepsilon(N),$$

where for $i \in \{0, 1\}$, p_i is the probability that output$(\mathcal{C}) = 1$ on a random run of $\mathcal{E}_i$. But p_i is the expected value of $p_i(n)$, where n is a randomly-chosen N-admissible number. It follows that for at least $\varepsilon(N)/2$ of the N-admissible numbers n, we have

$$|p_0(n) - p_1(n)| > \varepsilon(N)/2.$$

Given inputs n and z, $\mathcal{B}$ works by repeatedly "simulating" random elections in the systems $\mathcal{E}_0$ and $\mathcal{E}_1$ except that instead of the government randomly choosing its parameters n and y in phase 1 of $\mathcal{S}_0$, the simulator forces them to particular values: n is set to the value of the input "n" to the residue tester, and parameter y is set to $z^i x^r$ for randomly chosen i and x satisfying $0 < i < r$, $0 \leq x < n$, and $\gcd(x, n) = 1$. y has the property that it is an r^{th} residue modulo n if and only if z is. However, it can be shown that y is chosen uniformly from among the set of all r^{th} residues (respectively non-r^{th} residues) which are relatively prime to n. Call a simulated election using parameters derived from n and z as above an (n, z)-*pseudo election.*

Let $q_i(n, z)$ be the probability that $\mathcal{C}$ outputs 1 on an (n, z)-pseudo election of system $\mathcal{E}_i$. If z is an r^{th} residue, a symmetry argument shows that

$$q_0(n, z) = q_1(n, z),$$

for then the ballots of the honest voters consist of pairs of random r^{th}-residues in both $\mathcal{E}_0$ and $\mathcal{E}_1$, and the two runs are totally indistinguishable to processes in C. On the other hand, if z is a non-r^{th} residue, then $q_i(n, z) \approx p_i(n)$, so if n is such that $|p_0(n) - p_1(n)| > \varepsilon(N)/2$, then

$$|q_0(n, z) - q_1(n, z)| > \varepsilon'(N),$$

where ε' is another inverse polynomial function of N. Hence, whether or not z is an r^{th} residue can be determined with high probability by using statistical sampling to distinguish the case that $q_0(n, z) = q_1(n, z)$ from the case that $|q_0(n, z) - q_1(n, z)| > \varepsilon'(N)$. It follows using standard techniques of probability theory that by running polynomially-many (n, z)-pseudo elections, these two cases can be distinguished with error that goes to zero with increasing N more rapidly than any inverse polynomial function.

We still have said little about how the simulation is carried out. It is not simply a matter of simulating the steps of the individual processors in $\mathcal{E}_i$, for the simulator has two problems.

First, we only want $\mathcal{B}$ to simulate elections in which the government parameters selected in phase 1 are the partciular values n and y, for otherwise we gain no information about whether or not z is an r^{th} residue modulo n. Hence, rather than simulate truely random elections, $\mathcal{B}$ manipulates the beacon to insure that these parameters are the chosen ones. It does this in a way that still looks random to the other processors by choosing in step 1a a random number m' between 1 and η_1 and forcing $n_{m'} = n$ and $y_{m'} = y$, and then in step 1b forcing m, the number that would be chosen by the beacon in a real run of $\mathcal{E}_i$, to be m'.

The second problem is that the simulator does not know the factorization of n, yet step 4c seems to require that information. Examination of the protocol, however, will show that it suffices to know the values of the r^{th} roots used to generate all votes cast, for this enables the simulator to compute t directly, and knowing t, to compute x^r. Although it is still unable to find x, it can successfully simulate phase 4 in the following unorthodox way:

- Generate η_4 random bits b_i, $1 \leq i \leq \eta_4$.
- For each i, $1 \leq i \leq \eta_4$, randomly select a number d_i such that $\gcd(d_i, n) = 1$. If $b_i = 1$, then let $C_i = d_i^r$. If $b_i = 0$, then let $C_i = d_i^r / x^r$.

- Simulate step 4a by revealing the numbers C_i.
- Simulate step 4b by using the b_i as the beacon values.
- Simulate step 4c by revealing $(t, |\Gamma| - t)$ and the numbers d_i.

Thus, a run of $\mathcal{E}_0$ or $\mathcal{E}_1$ can be simulated by $\mathcal{B}$ if it is able to extract the r^{th} roots used to generate all votes cast. $\mathcal{B}$ itself chose the roots (in step 2a) for all of the voters in H, so it remains only to extract the roots for the votes cast by processes in $\mathcal{C}$ that voted properly. (These correspond with high probability to the processes $j \in C$ for which $check_V(j) = \mathsf{good}$.) To do this, $\mathcal{B}$ "re-runs" portions of the election using different beacon values at each of the steps where a process in C must interactively prove that its ballot is valid (that is, when it is posting the messages that would be posted by a proper voter in phase 2). $\mathcal{B}$ tries to find a second run which has $check_V(j) = \mathsf{good}$ and which agrees with the first run up to the posting of the numbers $(f_i^r \bmod n)$ and $(yg_i^r \bmod n)$ of step 2a but differs in some beacon bit b_i in step 2b. In one of the runs, f_i and $g_i \pmod n$ will appear on the bulletin board (corresponding to what a proper voter would do in step 2c), and in the other, the bulletin board will have $f_i \cdot f_0^{-1}$ and $g_i \cdot g_0^{-1} \pmod n$. Dividing corresponding values yields f_0 and g_0.

$\mathcal{B}$ succeeds in the simulation if it finds the roots for all $j \in C$ for which $check_V(j) = \mathsf{good}$. It will not always succeed, for it might turn out, for example, that $check_V(j) = \mathsf{bad}$ in every run except the given one. However, through a fairly lengthly analysis, which we defer to the full paper, it can be argued that such bad cases are relatively infrequent and that $\mathcal{B}$ succeeds often enough to satisfy our theorem.

This yields:

Theorem 2 *If there is no expected polynomial time weak r^{th} residue tester, then $\mathcal{S}_0$ is secure.*

6 Extensions, Variations, and Conclusions

6.1 Multiway Elections

We have considered only elections with two possible choices—"yes" and "no". The scheme presented extends easily to elections where many choices are allowed. In a three-way election, for example, the r used can be the product of two (known) primes r_1 and r_2. Votes could then be either of the form x^r, yx^r, or $y^{r_1}x^r$. As long as r_1 and r_2 are both greater than the maximum possible number of voters, the product of the votes will be uniquely expressible in the form $y^t x^r \pmod n$ for $0 \le t < r$, where t in turn is uniquely expressible as $t = ar_1 + b$ for $0 \le b < r_1$. a and b respectively denote the number of votes of the forms $y^{r_1}x^r$ and yx^r. The remaining valid votes are of the form x^r.

6.2 Eliminating the Beacon

The beacon used in these protocols can be replaced by a small set of "tellers" under the assumption that each voter trusts at least one of the tellers. The tellers together generate a random bit by each generating a random bit, releasing a one-way encryption of the bit, and then (when all encrypted bits have been released), releasing the actual (unencrypted) bit. The bits are then XORed to produce a single pseudo-random bit. This bit is random as long as any one teller's bit is itself random. A teller can, by refusing to release the decryption of a bit at some stage, bias the simulated beacon bit. If, however, improper tellers are excluded from further participation, the number of tellers is small, and the group of beacon bits in the process of being produced when the bad teller was exposed are recomputed from the beginning, then it can be shown that the dishonest tellers' influence is not sufficient to hinder the accuracy and security properties of the scheme.

One means of encrypting a bit for use by the teller's protocol is to generate an r^{th} residue modulo some n to indicate a 1 and a non-r^{th} residue to indicate a 0. This has the feature that security of the entire election remains dependent solely upon the difficulty of deciding r^{th} residues.

6.3 Related Schemes

In addition to the election scheme presented in this paper, a number of related schemes which all conform to the election paradigm of Figure 1 can be devised. One of these schemes is based upon the difficulty of computing the discrete logarithm modulo a known prime ([Adl79], [PoHe78], [CLS85]). Another is based upon the difficulty of determining the order of an element modulo the product of two unknown primes. A third proposed scheme encrypts a vote using the low order bit of a message encrypted by RSA [RSA78]. The major requirement of such a scheme seems to be the existance of a function that can be computed on a set of encrypted data which preserves the sum of the unencrypted components (see [RAD78]), but we do not yet know if that is sufficient for the correctness and security properties to carry over in general, nor have we

even investigated these particular examples carefully. Nevertheless, the variety of number theoretic problems on which such schemes can be based and the ease with which they have been found gives additional reason to believe in the usefulness of the general paradigm.

The notion of verifiability through generalized interactive proofs, as used in the election scheme presented here, seems to be quite powerful and may find application in other kinds of protocols as well.

6.4 Open Problems

One major disadvantage of our election paradigm is the government's powerful ability to read any vote. There are many applications in which a higher level of security is required. One way to achieve this, as suggested earlier, is to implement the government's process as a carefully monitored black box.

Eliminating altogether the government's ability to read individual's votes in a situation in which the number of voters is not known in advance, however, seems to be extremely difficult. For example, if the government is able to compute a tally for an arbitrary subset of the eligible voters, then it can compute a tally with and without the vote of a particular voter and thus determine that voter's vote. Thus, any such scheme must somehow restrict the government's ability to compute tallies to the set of voters who actually cast a vote in the election. We leave open the problem of finding such a scheme or proving its non-existence.

One possible tactic which may alleviate this problem in practice is to distribute the government's powers. With such a mechanism, it might be required that all parties which share the power cooperate in order to compute a tally. In such a case, it might be possible to maintain privacy under the assumption that the parties do not all collaborate to compromise the secrecy of an individual vote. No such mechanism for generalizing the paradigm presented here is yet known, but this may provide a fruitful avenue for further research.

References

[Adl80] Adleman, L. "On Distinguishing Prime Numbers from Composite Numbers." *Proc.* 21st *IEEE Symp. on Foundations of Computer Science*, Syracuse, NY (Oct. 1980), 387–406.

[Adl79] Adleman, L. "Subexponential Algorithm for The Discrete Logarithm Problem." *Proc.* 20th *IEEE Symp. on Foundations of Computer Science*, San Juan, PR (Oct. 1979), 55–60.

[AdMc82] Adleman, L. and McDonnell, R. "An Application of Higher Reciprocity to Computational Number Theory." *Proc.* 23rd *IEEE Symp. on Foundations of Computer Science*, Chicago, IL pp. (Nov. 1982), 100–106.

[APR83] Adleman, L., Pomerance, C., and Rumley, R. "On Distinguishing Prime Numbers from Composite Numbers." *Annals of Math. 117*, (1983), 173–206.

[Ben81] Ben-Or, M. "Probabilistic Algorithms in Finite Fields." *Proc.* 22rd *IEEE Symp. on Foundations of Computer Science*, Nashville, TN pp. (Oct. 1981), 394–398.

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

[CLS85] Coppersmith, D., Liscov, A., and Schroeppel R. "Discrete Logarithms in $GF(p)$." to appear in *Algorithmica*.

[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.

[DiHe76] Diffie, W. and Hellman, M. E. "New Directions in Cryptography." *IEEE Trans. on Information Theory 22*, 6, (Nov. 1976), 644–654.

[FMR84] Fischer, M., Micali, S., and Rackoff, C. "A Secure Protocol for the Oblivious Transfer." Presented at *Eurocrypt84*, Paris, France (Apr. 1984). (Not in proceedings.)

[GM82] Goldwasser, S. and Micali, S. "Probabilistic Encryption & How to Play Mental Poker, Keeping Secret All Partial Information." *Proc.* 14th *ACM Symp. on Theory of Computing*, San Francisco, CA (May 1982), 365–377.

[GM83] Goldwasser, S. and Micali, S. "Proofs With Untrusted Oracles." *Unpublished Manuscript* (May 1983).

[GMT82] Goldwasser, S., Micali, S., and Tong, P. "Why and How to Establish a Private Code On a Public Network." *Proc.* 23rd *IEEE*

Symp. on Foundations of Computer Science, Chicago, IL (Nov. 1982), 134–144.

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

[Kra84] Kranakis, E. "Theoretical Aspects of the Security of Public Key Cryptography." Yale University Department of Computer Science Technical Report 341. (Sep. 1984).

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

[PoHe78] Pohlig, S. and Hellman, M. "An Improved Algorithm for Computing Logarithms Over GF(2) and Its Cryptographic Significance." *IEEE Trans. on Information Theory 24*, 1 (Jan. 1978), 106–110.

[Rab79] Rabin, M. "Digitalized Signatures and Public-key Functions as Intractable as Factorization." MIT/LCS/TR-212. MIT Technical Report (Jan. 1979).

[Rab80] Rabin, M. "Probabilistic Algorithms in Finite Fields." *SIAM Journal on Computing 9*, 2 (May 1980), 273–280.

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

[RAD78] Rivest, R. L., Adleman, L., and Dertouzos, M.L. "On Data Banks and Privacy Homomorphisms." *Foundations of Secure Computation*, ed. by R. A. DeMillo, et. al. Academic Press, New York, 1978, 169–179.

[RSA78] Rivest, R. L., Shamir, A., and Adleman, L. "A Method for Obtaining Digitial Signatures and Public-key Cryptosystems." *Comm. ACM 21*, 2 (Feb. 1978), 120–126.

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