454 This work was supported in part by the National Security Agency under Grant MDA904–84–H–0004.

454 This work was supported in part by the National Security Agency under Grant MDA904–84–H–0004.

Improving Privacy in Cryptographic Elections

Josh D. Cohen

April 26, 1994

Abstract

This report describes two simple extensions to the paper *A Robust and Verifiable Cryptographically Secure Election Scheme* presented in the 1985 Symposium on the Foundations of Computer Science. The first extension allows the "government" to be divided into an arbitrary number of "tellers". With this extension, trust in any one teller is sufficient to assure privacy, even if the remaining tellers conspire in an attempt to breach privacy. The second extension allows a government to reveal (and convince voters of) the winner in an election *without* releasing the actual tally. Combining these two extensions in a uniform manner remains an open problem.

1 Introduction and Background

In [CoFi85], a protocol was presented which gives a method of holding a mutually verifiable secret-ballot election. The participants are the voters, a government, a trusted "beacon" which generates publically readable random bits, and a trusted global clock.

The protocol has four basic phases.

In phase 1, the government "calls for an election". In doing so, the government releases cryptographic parameters to be used in the election and engages in an interactive proof to give high confidence that these parameters meet certain specifications.

In phase 2, voters "register" for the election. This phase consists of each voter preparing a blank (unmarked) ballot and engaging in an interactive proof to give very high confidence that the ballot consists of one **no** vote and one **yes** vote.

In phase 3, voters "cast" their votes. Each registered voter has the opportunity to designate one of the two votes on his or her ballot as the vote to be cast.

In phase 4, the government announces the tally and releases additional information to substantiate the accuracy of the tally. With the aid of a final interactive proof, very high confidence is given that the tally released correctly represents the total number of **no** votes and the total number of **yes** votes cast.

For completeness, the protocol is given in figure 1.1.

Proofs are given in [CoFi85] that this protocol satisfies a correctness theorem and a privacy theorem. The correctness theorem states roughly that if the election passes a simple *check* function, then with high probability the tally claimed by the government is the desired tally of votes. The privacy theorem state roughly that, based on a number-theoretic assumption, no set of conspiring voters can distinguish between the possible votes of any other voters. The function *check* simply verifies that the government steps appear to be consistent with the government having followed its protocol and includes in the tally precisely the votes of voters who took actions which appear consistent with their protocols.

Perhaps the greatest shortcoming of the privacy theorem is that it excludes the government. That is to say, the government can tell exactly how every voter votes in every election. In section 2, a generalization to the original protocol is presented in which a set of "tellers" replace the government. The sum of the results obtained by the tellers represents the tally of an election; however, unless *all* tellers conspire, none will be able to determine how any individual voter voted. This technique is then further generalized to allow the computation of an accurate tally even in the presence of faulty tellers. A simulation technique is also discussed whereby the beacon is replaced by a set of election "officials".

In section 3, a second generalization is given whereby the government can announce (and convince the voters of) the winner of an election without releasing any details about the actual tally. This technique can also be used with the teller method above; however it seems to be necessary for at least one teller to know the exact tally of the election.

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, $1 \leq m \leq \eta_1$.

1c. For all $i \neq m$, reveal length N primes p_i and q_i such that $n_i = p_i q_i$, $r | (p_i - 1)$, and $r \nmid (q_i - 1)$. Denote (n_m, y_m) by (n, y).

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

2a. For $0 \leq i \leq \eta_2$, 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 $b_i, 1 \leq i \leq \eta_2$.

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 $b_i, 1 \leq i \leq \eta_4$.

4c. Compute x and t such that $W \equiv y^t x^r \pmod{n}$ and $0 \leq t < r$. 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 1.1: The basic election scheme $\mathcal{S}_1$.

2 Replacing the Government with Tellers

The basic idea of this generalization is to have each teller conduct a sub-election with all of the voters. A voter may cast any number of votes in any of the sub-elections with only one constraint. The total number of votes cast by any voter over all sub-elections must be either $0 \pmod r$ – indicating a **no** vote – or $1 \pmod r$ – indicating a **yes** vote. A form of interactive proof (see [GMR85], [CoFi85]) is used to ensure that this constraint is met.

This protocol denoted by scheme $\mathcal{S}_2$ is given in the four phases detailed in figures 2.1–2.4. Before the first phase, a prime r is fixed such that r is greater than the number of eligible voters. η_1, η_2, and η_4 are preset integers which depend on the security parameter N. An integer n is said to be N-admissible if $n = pq$ where p and q are primes of length N and where $r|p-1$ and $r\not|q-1$.

In phase 1, given in figure 2.1, each of the k tellers t_j selects parameter sets to be used in the election. Interactive proofs are then used to select one parameter set for each teller and give high confidence that these distinguished parameter sets are of the required form. This phase is no different from phase 1 of the basic election scheme $\mathcal{S}_1$, except that this phase is repeated for each teller.

Phase 1 steps executed by each teller $t_j \in T$:

1a. Release η_1 randomly chosen pairs (n_{j_i}, y_{j_i}), $1 \leq i \leq \eta_1$, such that $n_{i,j}$ is N-admissible, $\gcd(y_{i,j}, n_{i,j}) = 1$, and there exists no $x_{i,j}$ such that $x_{i,j}^r \equiv y_{i,j} \pmod{n_{i,j}}$.

1b. Use beacon/officials to generate a random integer m, $1 \leq m \leq \eta_1$.

1c. For all $i \neq m_j$, reveal length N primes p_{j_i} and q_{j_i} such that $n_{j_i} = p_{j_i} q_{j_i}$, $r|(p_{j_i} - 1)$, and $r\not|(q_{j_i} - 1)$. Denote (n_{j_m}, y_{j_m}) by (n_j, y_j).

Figure 2.1: The parameter setting phase of $\mathcal{S}_2$.

Recall that a pair of election parameters (n_j, y_j) was selected by each teller t_j in phase 1. A vector consisting of k integers which is expressible in the form $\langle y_1^{e_1} x_1^r, y_2^{e_2} x_2^r, \ldots, y_k^{e_k} x_k^r \rangle$ such that $\sum_j e_j \equiv C \pmod r$ denotes a C-vote. As before, a 0-vote represents a **no** vote, a 1-vote represents a **yes** vote, and a valid ballot is a pair consisting of a **no** vote and a **yes** vote. In phase 2 of the election, presented in figure 2.2, each voter prepares a single master ballot and many scratch ballots and engages in an interactive proof to demonstrate with very high confidence that the master ballot is valid.

The componentwise product (the j^{th} product taken modulo n_j) of a C_1-vote and a C_2-vote is easily seen to be a (C_1+C_2)-vote. Thus, the product of up to $r-1$ **no** votes and **yes** votes is a t-vote, where t is the number of votes which were **yes** votes. Therefore, the tally of an election is determined by the vote type of the the product of all legitimate votes cast in the election.

Note also that it is possible to demonstrate that two votes are of the same type by showing that their quotient is a 0-vote. This will be required in order to enable the *check* function to verify that a ballot is valid.

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

2a. For $0 \le i \le \eta_2$, randomly select vectors $\alpha_i = \langle \alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,k} \rangle$ and $\beta_i = \langle \beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,k} \rangle$ such that for all i and j, $0 \le \alpha_{i,j}, \beta_{i,j} < r$ and for all i, $\sum_j \alpha_{i,j} \equiv 0 \pmod r$ and $\sum_j \beta_{i,j} \equiv 1 \pmod r$.
For $0 \le i \le \eta_2$, randomly select vectors $f_i = \langle f_{i,1}, f_{i,2}, \ldots, f_{i,k} \rangle$ and $g_i = \langle g_{i,1}, g_{i,2}, \ldots, g_{i,k} \rangle$ such that $\gcd(f_{i,j}, n_j) = \gcd(g_{i,j}, n_j) = 1$ for all i and j.
For $0 \le i \le \eta_2$, let $F_i = \langle y_1^{\alpha_{i,1}} f_{i,1}^r, y_2^{\alpha_{i,2}} f_{i,2}^r, \ldots, y_k^{\alpha_{i,k}} f_{i,k}^r \rangle$ and $G_i = \langle y_1^{\beta_{i,1}} g_{i,1}^r, y_2^{\beta_{i,2}} g_{i,2}^r, \ldots, y_k^{\beta_{i,k}} g_{i,k}^r \rangle$, and release a ballot B_i consisting of the two vectors F_i and G_i in random order – (F_i, G_i) or (G_i, F_i).

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

2c. For all i such that $b_i = 1$, reveal α_i, β_i, f_i and g_i.
For all i such that $b_i = 0$, reveal $\alpha_i - \alpha_0 \pmod r$ and $\beta_i - \beta_0 \pmod r$; and for those j for which $\alpha_{0,j} \le \alpha_{i,j}$ (resp. $\beta_{0,j} \le \beta_{i,j}$) reveal $f_{i,j}/f_{0,j}$ (resp. $g_{i,j}/g_{0,j}$), otherwise reveal $y^{-1} f_{i,j}/f_{0,j}$ (resp. $y^{-1} g_{i,j}/g_{0,j}$).

Figure 2.2: The ballot preparation phase of $\mathcal{S}_2$.

Phase 3 consists of each voter actually designating which of his or her two votes is to be cast in the election. This phase is analogous to the marking of a (previously) blank ballot. Note that phases 1 and 2 may be completed well before the actual voting takes place. These phases may be viewed as a registration process, leaving the vote casting of phase 3 for "election day". See figure 2.3.

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

3. Select one element of B_0 as the actual vote w. To vote "no", select $w = F_0$. To vote "yes", select $w = G_0$.

Figure 2.3: The vote casting phase of $\mathcal{S}_2$.

The final phase of the election comprises the computation and release of the election tally. Each teller releases the tally of its sub-election as well as additional information which gives very high confidence that this sub-tally is accurate. Figure 2.4 shows the details of this phase. Like phase 1, this phase is unchanged from the basic election scheme $\mathcal{S}_1$, again with the exception that this phase is repeated for each teller.

Phase 4 steps executed by each teller $t_j \in T$:

4a. Compute $\Gamma = \{i : \mathit{check}_V(i) = \mathsf{good}\}$ and let $W_j = \prod_{i \in \Gamma} w_{i,j} \bmod n_j$ where $w_{i,j}$ is the j^{th} component of the vote of voter v_i. Randomly select η_4 numbers $c_{i,j}$ such that $\gcd(c_{i,j}, n) = 1$ and reveal all $C_{i,j} = c_{i,j}^r$.

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

4c. Compute x_j and τ_j such that $W_j \equiv y_j^{\tau_j} x_j^r \pmod{n}_j$ and $0 \le \tau_j < r$. Reveal τ_j. For all i such that $b_i = 1$, reveal $c_{i,j}$. For all i such that $b_i = 0$, reveal $c_{i,j} x_j$.

Figure 2.4: The tally tabulation phase of scheme $\mathcal{S}_2$.

The tally of the election (number of **yes** votes) is $\sum_j \tau_j \bmod r$.

The function *check*, as in the basic scheme, simply verifies that the actions taken by each teller are superficially consistent with those actions dictated by the protocol. It should be emphasized, however, that a voter's vote is not included in any sub-tally unless its vote in *all* sub-elections is consistent with the protocol.

The only subtle aspect of the *check* function is the verification that voter ballots are valid (step 2c). In the cases where the beacon bit b_i is 1, the disclosure of the corresponding f_i and g_i serve as proof that α_i and β_i are of the appropriate form and that the ballot B_i is therefore valid. In the cases where the beacon bit b_i is 0, the disclosure of the required quotients allow an observer to verify that $\alpha_i - \alpha_0$ and $\beta_i - \beta_0$ are each 0-votes. The sub-case in which the quotients are multiplied by y^{-1} normalizes the elements of $\alpha_i - \alpha_0$ and $\beta_i - \beta_0$ to the range from 0 to $r-1$. Together, these two cases give an observer extremely high confidence that each ballot B_i and in turn master ballot B_0 is valid.

The proof of correctness of the tally is almost identical to that in the basic protocol and is straightforward. The proof of privacy can be generalized under the assumption that at least one teller is honest (does not divulge its partial information to others). This generalization is again straightforward, since in the case where all but one teller collaborates, the remaining teller is roughly equivalent to an honest government.

The extreme cases are also of interest since the case of one teller is identical to that of the government in the basic scheme. The opposite extreme in which every voter is a teller guarantees privacy, but the robustness is lost since a faulty teller forces the election to be restarted (without that teller). This case is analogous to boardroom voting (see [DLM82],[Mer83],[Yao82]).

It should be emphasized that a faulty teller is used here to mean one that refuses to complete its protocol. Unless the failure occurs before ballots are prepared (allowing the election to be restarted easily), such a failure could be limited to a teller not releasing its required sub-tally. (This would be accomplished by moving the random number selection of step 4a into phase 1.) Since no partial information about the tally exists before this point, a simultaneous broadcast protocol (see [CGMA85], for example) can prevent an unhappy teller from refusing to release its sub-tally after seeing that the tally is not developing as it desires.

2.1 Recovering from Teller Faults

An alternative approach to dealing with the possibility of faulty tellers is given by embedding the entire election within a suitable secret sharing scheme. With such an embedding, it will be possible to hold an election with k tellers as in scheme $\mathcal{S}_2$. However, for a predetermined d, any subset consisting of d or more of the tellers can determine and prove to participants and observers the tally of the election. The disadvantage in this approach is that any subset consisting of d or more of the tellers can also determine how any individual voter voted in the election.

Although the entire election scheme is to be embedded within a secret sharing scheme, the modifications to scheme $\mathcal{S}_2$ which are required in order to achieve this embedding are surprisingly minor. Shamir's threshold scheme [Sha79] is particularly well suited for this purpose.

The key difference between the new scheme and scheme $\mathcal{S}_2$ is in the definition of a C-vote. Whereas a vector $\langle y_1^{e_1}x_1^r, y_2^{e_2}x_2^r, \ldots, y_k^{e_k}x_k^r \rangle$ was in scheme $\mathcal{S}_2$ interpreted as a C-vote if and only if $\sum_j e_j \equiv C \pmod{r}$, the same vector is interpreted in the modified scheme as a C-vote if and only if there is a polynomial of degree at most $d-1$ over the integers modulo r which passes through the points (j, e_j) for all j, $1 \leq j \leq k$ and which also passes through the point $(0, C)$.

The changes indicated by this modification can be enumerated as follows:

1. In step 2a of scheme $\mathcal{S}_2$, instead of requiring that each α_i and β_i be chosen such that $\sum_j \alpha_{i,j} \equiv 0 \pmod{r}$ and $\sum_j \beta_{i,j} \equiv 1 \pmod{r}$, we now require that each α_i and β_i be chosen such that the points $(j, \alpha_{i,j})$ together with the point $(0,0)$ all lie on a polynomial of degree at most $d-1$ over the integers modulo r and the points $(j, \beta_{i,j})$ together with the point $(0,1)$ all lie on a polynomial of degree at most $d-1$ over the integers modulo r.

2. The tally of the election, rather than being $\sum_j \tau_j \bmod r$, now becomes the value at 0 of the (at most degree $d-1$) polynomial (taken over the integers modulo r) defined by any d or more of the points (j, τ_j).

3. The function *check*, precisely as in scheme $\mathcal{S}_2$, must check that the information revealed in step 2c is consistent with the protocol. However, the votes revealed must be consistent with the new definition of a C-vote.

It might seem as though an adversarial group of less than d tellers could use its private information together with the information publically revealed by the remaining tellers to discern information about individual votes. It is seen in [Coh86], however, that this cannot happen.

There is an obvious trade off in the choice of d — the number of tellers required to complete the protocol. If d is chosen to be equal to k, then the properties achieved are the same as those achieved by scheme $\mathcal{S}_2$, although the approach is somewhat different. Here, trust in one teller is sufficient to assure privacy. In general, the number of tellers which must be trusted (in the sense that they will not conspire to compromise privacy) becomes $k - d + 1$. As d is reduced, $k - d + 1$ increases. An apparent limit is reached when d drops to half of k. At this point, only $k/2$ tellers need be trusted to complete the protocol. However, the remaining $k/2$ tellers, being untrustworthy, will presumably conspire to read individual votes. It seems that a d only slightly smaller than k — accounting for the possibility of a few saboteurs, and necessitating trust in only a few tellers — might be the best compromise.

2.2 Replacing the Beacon with Officials

The original paper briefly describes how the beacon can be replaced by a set of election officials (previously called tellers). By again using a simultaneous broadcast protocol such as [CGMA85], the private coins of officials can be revealed and XORed to form a secure public coin. For simplicity here it shall be assumed that a faulty official forces the election to be restarted (without that official), although a more complicated analysis seems to allow an election to be recovered and continued without the faulty official.

As with tellers, the use of officials can easily be incorporated into the correctness and privacy proofs. The assumption that at least one official is honest is now necessary for the correctness theorem. (Recall that the assumption of an honest teller was used for the privacy theorem.)

The extreme cases are again interesting. The case of one official is identical to that of a trusted beacon. The opposite extreme in which every voter acts as an official again causes a loss in robustness. It may be possible to retain the robustness in this latter case by repeating only a small portion of the election; however a complete analysis of the additional influence that may be gained by voters in this case has thusfar defied the author.

The case in which the tellers also serve as the officials is perhaps the simplest and most practical of the generalizations. In this case, trust in one teller/official gives all of the desired properties simultaneously.

3 Announcing the Winner without Divulging the Tally

In this section an extension to the basic scheme will be presented by which the winner of an election can be verified without giving any specific information about the actual tally. It may be desirable in some cases to allow the margin of victory in an election to be concealed, if for no other reason than to protect the self-esteem of the loser.

This can be accomplished by demonstrating that all possible tallies in favor of one candidate are *not* correct tallies, thus proving that this candidate is not the winner of the election. This involves establishing that up to $\frac{r}{2}$ specific integers are not r^{th} residues. In order to demonstrate non-residuosity, additional non-residues are prepared at the beginning of the protocol. Each known non-residue z can be used later to show that a subsequently derived integer w is also a non-residue by showing that w is expressible as $w \equiv z^i x^r \pmod{n}$ for integers x and i with $0 < i < r$.

The protocol of scheme $\mathcal{S}_3$ is shown in figures 3.1–3.4.

Figure 3.1 shows the parameter setting phase of scheme $\mathcal{S}_3$. The only difference between this and the first phase of the basic election scheme $\mathcal{S}_1$ is the inclusion of additional non-residues in each parameter set. The number of additional non-residues prepared must be large enough to accomodate the requirements of phase 4.

Phase 1 steps executed by government:

1a. Release η_1 randomly-chosen sets $(n_i, y_i, z_{i,0}, z_{i,1}, \ldots, z_{i,\lfloor r/2 \rfloor})$ such that n_i is N-admissible, $\gcd(y_i, n_i) = 1$, for all j, $\gcd(z_{i,j}, n_i) = 1$, and there exists no x_i such that $x_i^r \equiv y_i \pmod{n_i}$ or that $x_i^r \equiv z_{i,j} \pmod{n_i}$ for any j.

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

1c. For all $i \neq m$, reveal length N primes p_i and q_i such that $n_i = p_i q_i$, $r|(p_i - 1)$, and $r \not| (q_i - 1)$. Denote $(n_m, y_m, z_{m,0}, z_{m,1}, \ldots, z_{m,\lfloor r/2 \rfloor})$ by $(n, y, z_0, z_1, \ldots, z_{\lfloor r/2 \rfloor})$.

Figure 3.1: The parameter setting phase of scheme $\mathcal{S}_3$.

Figures 3.2 and 3.3 describe the ballot preparation and vote casting phases of scheme $\mathcal{S}_3$. These phases are unchanged from the basic election scheme $\mathcal{S}_1$. Voters paricipate in scheme $\mathcal{S}_3$ exactly as they do in $\mathcal{S}_1$, although additonal requirements are placed on the government.

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

2a. For $0 \leq i \leq \eta_2$, 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 $(yg_i^r \bmod n)$ in random order.

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

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 $g_i \cdot g_0^{-1} \bmod n$.

Figure 3.2: The ballot preparation phase of scheme $\mathcal{S}_3$.

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 $w_j = yg_0^r$. To vote "no", select $w_j = f_0^r$.

Figure 3.3: The vote casting phase of scheme $\mathcal{S}_3$.

The final phase, shown in figure 3.4, details the mechanism by which a possible tally can be shown to not be the correct election tally. Phase 4 repeats this process sufficiently often to demonstrate that the stated loser is the actual loser. Care must be taken not to re-use any of the pre-selected non-residues, since using one pre-selected non-residue to demonstrate the non-residuosity of more than one derived integer can have the side-effect of giving specific information about the tally. This is why $\lfloor r/2 \rfloor$, non-residues were prepared in phase 1.

4 Open Problems and Remarks

There are several components of the schemes seen here in which improvements are desired.

Phase 4 steps executed by government:

4a. Compute $\Gamma = \{j : check_V(j) = \texttt{good}\}$, $\Delta = \lfloor |\Gamma|/2 \rfloor$, and $W = \prod_{j \in \Gamma} w_j \bmod n$. For $1 \leq i \leq \eta_4$ and $0 \leq \delta \leq \Delta$, randomly select integers $c_{\delta,i}$ such that $\gcd(c_{\delta,i}, n) = 1$ and reveal all $C_{\delta,i} = c_{\delta,i}^r$.

4b. Use beacon to generate random bits $b_{\delta,i}$ such that $1 \leq i \leq \eta_4$ and $0 \leq \delta \leq \Delta$.

4c. Compute x and t such that $W \equiv y^t x^r \pmod{n}$ and $0 \leq t < r$. If $t = |\Gamma|/2$ announce *tie* and for all i such that $b_{0,i} = 1$, reveal $c_{0,i}$; for all i such that $b_{0,i} = 0$, reveal $c_{0,i}x$.
If $t > |\Gamma|/2$ announce *yes wins* and for all δ such that $0 \leq \delta \leq \Delta$ compute γ_δ and x_δ such that $0 < \gamma_\delta < r$ and $Wy^{-\delta} \equiv z_\delta^{\gamma_\delta} x_\delta^r \pmod{n}$. Reveal all γ_δ. For all i such that $b_{\delta,i} = 1$, reveal $c_{\delta,i}$. For all i such that $b_{\delta,i} = 0$, reveal $c_{\delta,i}x$.
If $t < |\Gamma|/2$ announce *no wins* and for all δ such that $\Delta \leq \delta \leq |\Gamma|$ compute γ_δ and x_δ such that $0 < \gamma_\delta < r$ and $Wy^{-\delta} \equiv z_\delta^{\gamma_\delta} x_\delta^r \pmod{n}$. Reveal all γ_δ. For all i such that $b_{|\Gamma|-\delta,i} = 1$, reveal $c_{|\Gamma|-\delta,i}$. For all i such that $b_{|\Gamma|-\delta,i} = 0$, reveal $c_{|\Gamma|-\delta,i}x$.

Figure 3.4: The tally tabulation phase of scheme $\mathcal{S}_3$.

4.1 Combining Schemes $\mathcal{S}_2$ and $\mathcal{S}_3$

It would be nice to be able to combine the two new schemes presented in this report. It is possible to substitute a set of officials for the beacon in scheme $\mathcal{S}_3$; however using tellers instead of the government does not seem to generalize well here. The problem seems to be that in order to show that a certain tally is not correct, it seems necessary that at least one party know what tally *is* correct.

If all tellers but one announce and publically prove their sub-tallies, then the remaining "master teller" could (with its private knowledge of its own sub-tally) engage in a protocol similar to that of scheme $\mathcal{S}_3$ and prove that its sub-tally does not correspond to any of the values which would make one candidate the winner, thereby proving that the other is the winner.

Perhaps each teller could begin by proving that one particular integer is *not* the sub-tally which it holds. After each teller has done this, they might next prove that an adjacent integer is also not its sub-tally. If this process is continued until each of k tellers holds only two possible sub-tallies (say t and $t+1$), then the election tally is constrained to one of $k+1$ consecutive values. How to ensure that these $k+1$ values all represent the same winner or to generalize this technique so that all tallies which give victory to the winning candidate are possible remains open.

4.2 Recovering from Faults

One of the major advantages of these election schemes is that a voter who does not comply (or at least seem to comply) with the election protocol can simply be ignored. By using a government and beacon and assuming their reliability, it was possible to eliminate the possibility of a failed election.

When tellers replace the government and officials replace the beacon, the problem of failures among these agents cannot be ignored. Faults here are taken to mean detectable faults — generally refusing to continue a protocol that has been begun. Undetected deviations from the prescribed protocol can only with very low probability affect the outcome of an election.

By simply declaring that a faulty teller or official causes the election to fail, the problem of faults may be dismissed easily. This is not entirely unsatisfactory, since the number of tellers and officials is presumed to be relatively small. A better solution would, however, be desirable.

In section 2.1, we examined how the techniques of secret sharing could be used to enable an election to be completed despite some predetermined number of faulty tellers. The problem of recovering an election with a faulty or malicious official seems to be somewhat more difficult.

The role of the officials is simply to generate random bits for use in interactive proofs: one interactive proof with each teller in each of phases 1 and 4, and one interactive proof with each voter in phase 2. It seems that if an official fails during any such proof step, then only that proof need be repeated (without the faulty official), and subsequent proofs can be executed with the faulty official excluded. All indications have thus far agreed with this intuition, but the proof that the influence exerted by an official choosing to stop is negligible has been excruciating and has not yet been completed.

4.3 Achieving Exponentially Strong Confidence

In phase 1 of the schemes presented, the probability of defeating the interactive proof decreases only linearly with the number of parameter sets chosen (doubling the number of parameter sets only halves the probability of cheating), whereas the interactive proofs in all subsequent steps of these schemes decrease exponentially with the number of scratch ballots or random integers chosen (each additional scratch ballot required halves the probability of a voter cheating successfully). Finding an exponentially secure interactive proof for phase 1 of these schemes is very much desired.

One possible solution to this problem would be to hold many simultaneous elections and to demonstrate that each results in the same tally. In order to meet this condition, voters would be required to cast the same vote (**yes** or **no**) in each election. The government would be required to reveal any voter who does not comply. This would ensure that every election does in fact yield the same tally.

This solution could also be applied when tellers are used instead of a government. Each voter would now be required to give the same sub-votes to each teller in each election (i.e. use the same α_0 or β_0 in each election — see figure 2.2). Each teller would then be responsible for exposing any voter who voted inconsistently in that teller's sub-election, and, once the inconsistency is verified, the votes submitted by that voter would be excluded from all sub-tallies.

This approach, however, represents a significant increase in the work necessary to hold an election, and it is not yet known how to generalize the privacy proof for this extension. A local version of an exponentially secure interactive proof would still be prefered for phase 1.

4.4 The Difficulty of Deciding r^{th} Residues

All of the schemes presented have relied on (in the sense that privacy is dependent on) the difficulty of distinguishing between residues and non-residues. This problem is believed to be hard, but more information as to its precise complexity would be very useful.

Adelman and McDonnell in [AdMc82] (see also [APR83]) have shown that the problem is *almost* as hard as factoring when the exponent r is allowed to vary. It would be nice to show that the fixed exponent case used here is as hard as factoring.

The problem of deciding r^{th} residues is certainly in $\mathcal{NP}$, but it would be *very* nice to show that it is $\mathcal{NP}$-complete — yielding among other things the corollary that factoring is $\mathcal{NP}$-hard. Manders and Adelman in [MaAd78] have shown that the problem is in fact $\mathcal{NP}$-complete (even when r is 2) when the size of the potential r^{th}roots is constrained. The reduction, however, is based on the large number of roots which might have to be considered and not on any intrinsic difficulty of distinguishing between residues and non-residues.

It would be *extremely* nice to show that the complexity of the problem is super-polynomial — thereby separating $\mathcal{P}$ and $\mathcal{NP}$, but there has thus far been only limited progress in this direction.

Finally, it would be nice to develop some kind of intractability result, but this is probably a bit much to ask.

Acknowledgements

Many debts are owed to people who have contributed to this work. Moti Yung developed the first generalization to the one government protocol. His idea allowed two governments to each hold an election such that the overall tally was the difference of the individual tallies. Adi Shamir originally suggested the problem of announcing a winner without the tally. He, together with Oded Goldreich and Ron Rivest developed the key idea which led to this generalization. Dana Angluin suggested holding simultaneous elections to achieve exponential security. Jerry Leichter, David Wittenberg, and Moti Yung have made many contributions to various portions of this work. Finally, my advisor, Mike Fischer has been instrumental in all phases of this (and of course the original) work.

References

[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.

[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.

[Coh86] **Cohen, J.** “Keeping Shares of a Secret Secret.” *TR-453, Yale University, Departement of Computer Science*, New Haven, CT (Feb. 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.

[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.

[MaAd78] **Manders, K.** and **Adleman, L.** “NP-Complete Decision Problems for Binary Quadratics.” *J. Comput. System Sci. 16*, (1978), 168–184.

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

[Sha79] **Shamir, A.** “How to Share a Secret.” *Comm. ACM 22*, 11 (Nov. 1979), 612–613.

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