Multi-Authority Secret-Ballot Elections with Linear Work

Ronald Cramer*
Matthew Franklin**
Berry Schoenmakers***
Moti Yung†

Abstract. We present new cryptographic protocols for multi-authority secret ballot elections that guarantee privacy, robustness, and universal verifiability. Application of some novel techniques, in particular the construction of witness hiding/indistinguishable protocols from Cramer, Damgård and Schoenmakers, and the verifiable secret sharing scheme of Pedersen, reduce the work required by the voter or an authority to a linear number of cryptographic operations in the population size (compared to quadratic in previous schemes). Thus we get significantly closer to a practical election scheme.

1 Introduction

An electronic voting scheme is viewed as a set of protocols that allow a collection of voters to cast their votes, while enabling a collection of authorities to collect the votes, compute the final tally, and communicate the final tally that is checked by talliers. In the cryptographic literature on voting schemes, three important requirements are identified:

Universal Verifiability ensures that any party, including a passive observer, can convince herself that the election is fair, i.e., that the published final tally is computed fairly from the ballots that were correctly cast.

Privacy ensures that an individual vote will be kept secret from any (reasonably sized) coalition of parties that does not include the voter herself.

Robustness ensures that the system can recover from the faulty behavior of any (reasonably sized) coalition of parties.

The main contribution of this paper is to present an **efficient** voting scheme that satisfies universal verifiability, privacy and robustness. Central to our methods is the application of witness indistinguishable protocols from [CDS94] to

* CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. cramer@cwi.nl
** AT&T Bell Labs., 600 Mountain Ave., Murray Hill, NJ 07974, USA. franklin@big.att.com
*** DigiCash bv, Kruislaan 419, 1098 VA Amsterdam, The Netherlands. Work done while at CWI. berry@digicash.com
† IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. moti@watson.ibm.com

Appears in: Ueli Maurer (ed.) EUROCRYPT'96, LNCS 1070, pp. 72–83.
Springer-Verlag Berlin Heidelberg 1996.

achieve greater simplicity and efficiency than previous voting schemes. The efficiency of our schemes can be summarized as follows, when there are n authorities, m voters, and security parameter k. The total amount of communication will be $O(kmn)$ bits (posted to a "bulletin board"), while the required effort (in elementary operations) for any authority and any voter will be $O(km)$ and $O(kn)$ operations, respectively. For any threshold $t \leq n$, privacy will be assured against coalitions that include at most $t-1$ authorities, and robustness against coalitions that includes at most $n-t$ authorities.

A fourth property recently stated is that of *vote-duplication*, i.e., one voter copying the vote of another voter without knowing the actual vote (see [SK95, Gen95]). There are various efficient ways to incorporate this property in the schemes we present. We will include a straightforward solution.

1.1 Overview of the Approach

The parties in a voting scheme are modelled as probabilistic polynomial time processes. Two means of communication are typically assumed to be available for these parties:

A **bulletin board**, which is a broadcast channel with memory that can be observed and read by all parties. Each party controls her own section of the board in the sense that she can post messages exclusively to her own section, but not to the extent that she can erase or overwrite previously posted messages.

Private channels to support private communication between voters and authorities. For this task any secure public-key encryption scheme is suitable, possibly using the bulletin board to post the corresponding encryptions.

The parties of the voting scheme perform the following steps to execute an election. To cast a vote, each voter constructs a ballot as an encryption of the desired vote, and posts the ballot to the bulletin board. At this point, a proof of validity is also required that convinces all parties that the posted encryption contains a valid vote, without revealing it. The authorities, however, are able to decrypt the ballots (because of extra information received from the voter through a private channel). In the end, the final tally is published together with some auxiliary information to enable universal verifiability: any interested party (a tallier) may "accumulate" the encrypted votes and check the final tally, by holding it against this accumulation and the auxiliary information.

More technically, universal verifiability is achieved by requiring the encryption function to be suitably homomorphic. At the same time a different security property of the encryption function, which is similar to the binding property of commitment schemes, ensures that the authority (assume momentarily to be a single entity) cannot accumulate the individual votes in any other way than the voters actually voted. Such homomorphic encryption schemes are available under a wide variety of common cryptographic assumptions.

Central to our results is the way we achieve an efficient proof of validity for ballots. The proof of validity shows to any interested party that a ballot actually represents a vote, e.g., that it either represents a yes or a no, and nothing else. To maintain privacy for the voters, the general idea is to use some sort of zero-knowledge proof. The problem is however that zero-knowledge proofs usually require a large number of repetitions before the desired level of confidence is achieved. The efficiency of these proofs to a great extent influences the efficiency of the whole scheme, both in terms of computational effort and in terms of the required bandwidth for each voter.

Our contribution now is twofold. We use a particular efficient homomorphic encryption scheme, based on the discrete logarithm problem (although, as we show, it can be based on other cryptographic assumptions as well). Moreover, by applying results from [CDS94], the proof of validity is a simple three-move protocol which is *witness indistinguishable* (in fact, witness hiding as well), instead of zero-knowledge as in previous schemes. This leads to a significant reduction of the effort required by the voter, from quadratic in the security parameter to linear, while still hiding the vote.

Clearly, in the scenario above, the authority learns individual votes. This situation is alleviated by having multiple authorities instead of one. The encrypted vote is distributed over the authorities such that fewer than some number of authorities remain ignorant about individual votes. To make sure that the authorities are convinced that the posted shares actually represent the vote cast, verifiable secret sharing is employed. Here, we apply Pedersen's scheme [Ped92], as it fits with the other primitives remarkably well. It is also by this method that robustness is achieved in the sense that only a subset (of a size larger than a certain threshold) of the authorities is required to participate throughout the execution of the scheme in order to compute the final tally.

1.2 Related Work

The type of voting schemes considered in this paper was first introduced and implemented in [CF85, BY86, Ben87b]. In these schemes, privacy and robustness are achieved by distributing the ballots over a number of tallying authorities, while still achieving universal verifiability. This contrasts with other approaches in which the ballots are submitted anonymously to guarantee privacy for the individual voters. Such schemes rely on the use of anonymous channels [Cha81], or even some form of blind signatures as in privacy-protecting payment systems (see, e.g., [Che94]) to achieve privacy. For these approaches it seems difficult however to attain all desired properties, and still achieve high performance and provable security.

The voting schemes of [CF85, BY86, Ben87b] rely on an r-th residuosity assumption. In [SK94] it is shown that such schemes can also be based on a discrete logarithm assumption (without fully addressing robustness, though), and how this leads to considerable efficiency improvements. In the present paper we will also address various number-theoretic assumptions.

As with our result, the efficiency improvement in [SK94] is mainly due to an improved zero-knowledge protocol to show validity of ballots. As noted by Benaloh, such *cryptographic capsules* [Ben87a, Ben87b] are at the heart of the problem of electronic elections and also useful for other applications. We note that the technique of [CDS94] can also be used to obtain efficient solutions for group signatures (see [CDS94, CP95]).

2 Cryptographic Primitives

2.1 The Discrete Logarithm Problem

Let $\mathcal{G} = \{\mathcal{G}_k\}$ be a family of groups of prime order such that the group operations can be performed efficiently, group elements can be efficiently sampled with uniform distribution and group membership as well as equality of group elements can be efficiently tested. Let *Gen* be a probabilistic polynomial time generator that on input 1^k outputs a description of a group $G \in \mathcal{G}_k$ (including the group order), and two random elements g, h from G. We say that the discrete logarithm problem for $\mathcal{G}$ is intractable (over *Gen*) if there is no probabilistic polynomial time algorithm that on input of G, g and h as output by $Gen(1^k)$ can compute $\log_g h$ with non-negligible probability in k.

Each family $\mathcal{G}$ for which it is reasonable to assume the intractability of the discrete logarithm problem is suitable for our purpose of constructing efficient and secure homomorphic encryption schemes with corresponding proofs of validity. A well-known family, however, is obtained by choosing large primes p and q at random such that $q \mid p - 1$; then G is the unique subgroup of order q in $\mathbb{Z}_p^*$. The discrete logarithm problem for elliptic curves over finite fields is also a candidate for implementation.

2.2 Homomorphic Encryption with Efficient Proof of Validity

Let $\mathcal{G}$ be a family of groups of prime order, and generator *Gen* as above. Assume that the discrete logarithm problem for $\mathcal{G}$ is intractable. The encryption scheme below is obtained as an extension of Pedersen's commitment scheme [Ped92] with an efficient proof of validity.

Initialization: The participants, or a designated subset of them, run $Gen(1^k)$ and obtain a group G_q of prime order q, and random group elements g and h. One way to do this honestly, is that the participants agree on a program for *Gen* first. Then they each run separately $Gen(1^k)$ where the coinflips needed are selected mutually at random, either by observing a common source of physical randomness, or by executing some well-known cryptographic protocol suitable for this purpose.

Encryption: A participant encrypts $v \in \mathbb{Z}_q$ by choosing $\alpha \in \mathbb{Z}_q$ at random, and computing

$$B \leftarrow g^\alpha h^v.$$

Opening: A participant can later open B by revealing both v and α. A verifying party then checks whether $B = g^\alpha h^v$, and accepts v as the encrypted value.

Voter			Verifier
$v = 1$	$v = -1$		
$\alpha, r_1, d_1, w_2 \in_R \mathbb{Z}_q$	$\alpha, r_2, d_2, w_1 \in_R \mathbb{Z}_q$		
$B \leftarrow g^\alpha h$	$B \leftarrow g^\alpha / h$		
$a_1 \leftarrow g^{r_1}(Bh)^{-d_1}$	$a_1 \leftarrow g^{w_1}$		
$a_2 \leftarrow g^{w_2}$	$a_2 \leftarrow g^{r_2}(B/h)^{-d_2}$		
		$\xrightarrow{B, a_1, a_2}$	
			$c \in_R \mathbb{Z}_q$
		$\xleftarrow{c}$	
$d_2 \leftarrow c - d_1$	$d_1 \leftarrow c - d_2$		
$r_2 \leftarrow w_2 + \alpha d_2$	$r_1 \leftarrow w_1 + \alpha d_1$		
		$\xrightarrow{d_1, d_2, r_1, r_2}$	
			$d_1 + d_2 \stackrel{?}{=} c$
			$g^{r_1} \stackrel{?}{=} a_1 (Bh)^{d_1}$
			$g^{r_2} \stackrel{?}{=} a_2 (B/h)^{d_2}$

Fig. 1. Encryption and Proof of Validity of Ballot B

Homomorphic Property: Encryption is homomorphic in the sense that, if B_1 and B_2 are encryptions of v_1 and v_2, respectively, then $B_1 B_2$ is an encryption of $v_1 + v_2 \bmod q$. In general, an encryption of any linear combination of v_1 and v_2 can be obtained from B_1 and B_2; in particular, the sign of v_1 can be flipped by computing B_1^{-1}. Note also that, e.g., $B_1 h$ encrypts $v_1 + 1 \bmod q$.

Proof of Validity: In our voting scheme to follow, it will be the case that a voter posts an encryption of a value $v \in \{1, -1\}$. To demonstrate that the encrypted value is indeed in $\{1, -1\}$, without revealing it, the voter and the verifier execute the efficient *proof of validity* of Figure 1. This proof requires only a very small number of modular exponentiations.

Theorem 1 *Under the discrete logarithm assumption, the encryption scheme is binding in the sense that it is infeasible to open a given encryption in two different ways. Furthermore, the proof of validity is a convincing argument that a given encryption is indeed an encryption of a value from the set* $\{1, -1\}$*, thereby not releasing any information about the actual value.*

Proof If any party is able to open an encryption B in two different ways, i.e., to present values α, v, α', v' such that $B = g^\alpha h^v = g^{\alpha'} h^{v'}$ with $\alpha \neq \alpha'$ and $v \neq v'$, it follows that $\log_g h = \frac{\alpha - \alpha'}{v' - v}$, which contradicts the discrete logarithm assumption. Furthermore, by the results of [CDS94], the protocol in Figure 1 is a witness indistinguishable proof of knowledge that the voter knows $\log_g Bh$ or $\log_g B/h$; here, Schnorr's identification protocol [Sch91] is used as the basic protocol in the construction of [CDS94]. Thus the verifier learns that the voter knows α and $v \in \{1, -1\}$ such that $B = g^\alpha h^v$, without obtaining any information about the actual value of v. $\square$

Note that it even follows that the proof of validity is witness hiding.

Jumping ahead a little bit, envision that a voter posts an encryption of his vote and that all other participants must verify its validity. As depicted in Figure 1, a source of randomness is required in the program for the verifier. For this purpose, one can use some unpredictable physical source of randomness [CF85], or agree on mutually random bits by cryptographic means. A more practical way, however, making the protocol *non-interactive*, is to apply the well-known Fiat-Shamir heuristic [FS87]. Their idea is to compute the challenge in a three-move identification scheme as a hash value of the message to be signed and the first message of the prover. So, let $\mathcal{H}$ be a suitable strong cryptographic hash function (thought of as a random oracle). In the non-interactive version of our proof of validity, the challenge c is computed as $c = \mathcal{H}(B, a_1, a_2)$. The set of values d_1, d_2, r_1 and r_2 is denoted proof(B). Given the values in proof(B), any participant can check the validity of B by verifying that $d_1 + d_2 = \mathcal{H}(B, g^{r_1}(Bh)^{-d_1}, g^{r_2}(B/h)^{-d_2})$.

2.3 Verifiable Secret Sharing

To achieve robustness efficiently, non-interactive verifiable secret sharing is employed. We use the scheme of Pedersen [Ped92], as it is based on discrete logarithms as well and fits nicely with our encryption scheme. Being information theoretically secure, this scheme also contributes to privacy in our multiple authority scenario.

3 Secret Ballot Election Scheme

We now present our main result, a secret ballot election scheme satisfying privacy, universal verifiability and robustness. The participants in the election scheme are n authorities $A_1, \dots, A_n$ and m voters $V_1, \dots, V_m$. Privacy and robustness are as follows. No collusion of fewer than t authorities can reveal an individual vote, while the election will be successful when at least t authorities operate properly ($1 \le t \le n$). At the same time we incorporate a simple mechanism to postpone the decision on what to vote until the preparation of the election has been completed. In this way several elections can be prepared beforehand at the beginning of the year, say, and casting a vote in an election then boils down to publishing essentially one bit of information.

Informally the scheme works as follows. Each voter V_i prepares a vote by randomly selecting a number b_i in $\{1, -1\}$. The voter first encrypts b_i by computing $B_i = g^{\alpha_i} h^{b_i}$, where $\alpha_i \in \mathbb{Z}_q$ is chosen randomly, and posts B_i to the bulletin board. Subsequently, b_i is considered as a secret which is to be shared among the authorities. We employ a verifiable secret sharing scheme to prevent voters from disrupting elections by sending false shares to authorities. The efficient scheme by Pedersen [Ped92] is a perfect candidate as it applies exactly to the discrete log setting we are considering. The idea is thus to let the voter act as the dealer in Pedersen's scheme, sending a verifiable share of the secret b_i to each authority using the proper private channels. The voter also posts proof(B_i) to the bulletin board to prove that B_i indeed encrypts a value in $\{1, -1\}$. Later, voter V_i may then cast a vote $v_i \in \{1, -1\}$ by publishing the value $s_i = b_i v_i$. In the end, the aggregate value $T = \sum_{i=1}^{m} v_i$ reduced modulo q such that $-q/2 < T < q/2$ represents the total number of yes-votes minus the total number of no-votes, hence the total number of yes-votes is $(m + T)/2$. For these numbers to be correct the obvious requirement is that $m < q/2$.

We assume that the group G_q and the members g and h are generated as described in Section 2. In particular, it then follows that $\log_g h$ is not known to any participant.

Ballot Construction

Each voter V_i prepares a masked vote $b_i \in \{1, -1\}$ in the following way.

1. The voter chooses b_i randomly from $\{1, -1\}$, and computes the ballot $B_i = g^{\alpha_i} h^{b_i}$, where α_i is randomly chosen from $\mathbb{Z}_q$. The voter also computes proof(B_i). Finally, the voter determines polynomials G_i and H_i,

$$G_i(x) = \alpha_i + \alpha_{i1} x + \cdots + \alpha_{i,t-1} x^{t-1}$$
$$H_i(x) = b_i + \beta_{i1} x + \cdots + \beta_{i,t-1} x^{t-1},$$

   where the coefficients α_{il}, β_{il}, $1 \leq l < t$, are chosen at random from $\mathbb{Z}_q$. Also, for these coefficients the voter computes the commitments $B_{il} = g^{\alpha_{il}} h^{\beta_{il}}$.
2. The voter posts B_i, proof$(B_i), B_{i1}, \ldots, B_{i,t-1}$ to the bulletin board.
3. All participants verify whether ballot B_i is correctly formed by checking proof(B_i).[5]
4. The voter sends the respective shares $(a_{ij}, b_{ij}) = (G_i(j), H_i(j))$ to authority A_j, using a private channel.
5. Each authority checks the received share (a_{ij}, b_{ij}) by verifying that

$$g^{a_{ij}} h^{b_{ij}} = B_i \prod_{l=1}^{t-1} B_{il}^{j^l}.$$

Vote Casting

To cast a vote, V_i simply posts $s_i \in \{1, -1\}$ such that $v_i = b_i s_i$ represents the desired vote.

[5] To prevent vote duplication, a bit string specific to voter V_i is also included in the input to the hash function $\mathcal{H}$ in proof(B_i). In case the proof of validity is done interactively, as depicted in Figure 1, this bit string could be incorporated in the challenge.

Tallying

1. Each authority A_j posts the sum $S_j = \sum_{i=1}^{m} a_{ij} s_i$ and the sub-tally $T_j = \sum_{i=1}^{m} b_{ij} s_i$.
2. Each tallier checks the share (S_j, T_j) posted by authority A_j by verifying that
$$g^{S_j} h^{T_j} = \prod_{i=1}^{m} \left(B_i \prod_{l=1}^{t-1} B_{il}^{j^l} \right)^{s_i} .$$
3. From t pairs (j, T_j) that correspond to authorities for which the shares (S_j, T_j) are correct, each tallier can compute the final tally T from the formula:
$$T = \sum_{j \in A} T_j \prod_{l \in A \setminus \{j\}} \frac{l}{l - j},$$
where A denotes a set of t correct authorities.

We assume (w.l.o.g.) that in a successful election the shares of every voter have been accepted by all authorities. That is, all verifications by the authorities in the last step of the ballot construction are successful. In case an authority receives a share that does not pass this step, the authority may post the share so that anybody can verify that the share is not correct and that it corresponds to the posted encryption of step 4 of the ballot construction.

Theorem 2 *Under the discrete logarithm assumption, our secret-ballot election scheme satisfies universal verifiability, robustness and privacy.*

Proof To prove universal verifiability first note that only correct ballots are counted on account of Theorem 1. Further, to prove that the final tally is correct, we reason as follows for each correct authority A_j. Let $G(x) = \sum_{i=1}^{m} s_i G_i(x)$ and $H(x) = \sum_{i=1}^{m} s_i H_i(x)$. By the binding property[6] of the encryptions B_i (see Section 2), we have:

$$\begin{aligned} g^{G(j)} h^{H(j)} &= g^{\sum_{i=1}^{m} s_i G_i(j)} h^{\sum_{i=1}^{m} s_i H_i(j)} \\ &= g^{\sum_{i=1}^{m} s_i a_{ij}} h^{\sum_{i=1}^{m} s_i b_{ij}} \\ &= \textstyle\prod_{i=1}^{m} (g^{a_{ij}} h^{b_{ij}})^{s_i} \\ &= \textstyle\prod_{i=1}^{m} (B_i \prod_{l=1}^{t-1} B_{il}^{j^l})^{s_i}. \end{aligned}$$

By the assumption that the verification in step 2 of the tallying protocol holds for (S_j, T_j), we thus have $g^{G(j)} h^{H(j)} = g^{S_j} h^{T_j}$, which implies $S_j = G(j)$ and $T_j = H(j)$ under the discrete logarithm assumption. As a consequence, the final tally T is indeed equal to $H(0)$ and thus represents the result of the election

[6] The description of our scheme above assumes that the hash function in the Fiat-Shamir style proof(B_i), behaves like a random-oracle. Instead of using this technique, the participants can get the necessary random bits as explained in Section 2, thus also removing the need for this extra assumption.

if the verification in step 2 of the tallying holds for at least t authorities. This deals with universal verifiability and robustness. The privacy property can easily be proved from the fact that the secret sharing scheme used and the proofs of validity (see Section 2) are information-theoretically secure. $\square$

Thus, fewer than t authorities do not obtain any information about individual votes, other than what can be derived from the final tally (and accounting for votes that have been revealed by individual voters).

To summarize the *performance* of our scheme, we note the following. The hard operations are the modular exponentiations with full exponents (i.e., exponents of expected size $|q|$). Counting these operations, we see that the work for each voter is $O(1)$ for the construction of the ballot (including the proof of validity) plus $O(t)$ for the commitments for the verifiable secret sharing scheme. Each authority has to do $O(m)$ work to check all the shares. So active participants do linear work. Finally, verification of the election (done only by interested talliers) requires $O(m)$ work to check the ballots plus $O(t \log^2 t)$ multiplications to check the shares of the final tally. Note that the parties' work can all be done either before or after the casting of the votes. The actual election simply consists of every voter posting essentially one bit of information to the bulletin board. This represents essentially an improvement of two orders of magnitude over the best schemes known so far (assuming $k = 100$), and enables much of the work required to be done off-line.

4 Extensions

The ballots B we have worked with so far are basically of the form $B = g^\alpha h^b$, where b represents the (masked) vote. There are numerous ways to extend this form, which will be elaborated upon in forthcoming work. We sketch some of the ideas in this section.

Parallel elections Several elections may be held at the same time by running several instances of the scheme from Section 3 in parallel. A possibly more efficient approach to perform K elections in parallel is to use ballots B of the form $B = g^\alpha h_1^{v_1} \cdots h_K^{v_K}$, where each v_l denotes the intended vote for the l-th election. The scheme of Section 3 can be carried over easily to this setting by observing that the immediate generalization of Okamoto's variation of Schnorr's scheme [Oka93] can be used to prove knowledge of either $\log_{g,h_1,\ldots,h_{l-1},h_{l+1},\ldots,h_K} Bh_l$ or $\log_{g,h_1,\ldots,h_{l-1},h_{l+1},\ldots,h_K} B/h_l$, for each $l = 1, \ldots, K$. Again the technique from [CDS94] can be applied to hide which of the two is known.

Multiway Elections In a multiway election each voter has to choose between a number of options. That is, if there are K options, the voter has to say yes to one of them and no to all of the others. (See also [BY86].) This is achieved efficiently through ballots of the same form as in a parallel election, with the additional requirement that $\sum_{l=1}^{K} v_l = 1$, if yes and no are represented by 1 and 0 respectively.

An even more efficient way is to take $B = g^\alpha h^{M^l}$ to represent option l, where M is larger than the number of voters. The voter then proves knowledge of at least one of $\log_g B/h, \log_g B/(h^M), \ldots, \log_g B/(h^{M^K})$, without revealing which. Note, however, that the number of options K is now bounded by approximately $\log_M q$.

Batched Elections A drawback of the above solutions for parallel and multi-way elections is that there is no mechanism to perform masked votes. An interesting extension therefore is to consider *batched* elections, where the same ballot is used several times. So, with a ballot of the form $B_i = g^{\alpha_i} h_1^{b_{i1}} \cdots h_K^{b_{iK}}$, a voter can later post the numbers s_{il} such that $v_{il} = b_{il} s_{il}$ represents the intended vote for the lth election, for $l = 1, \ldots, K$. Although this reveals the sums $\sum_i \alpha_i s_{il}$, $\sum_i b_{i1} s_{il}$, ..., $\sum_i b_{iK} s_{il}$, for $l = 1, \ldots, K$, this is not a problem if the number of voters is sufficiently large. (Note that $\sum_i b_{il} s_{il}$ is the result of the l-th election.)

Integer Votes Instead of requiring that votes are in $\{0, 1\}$ say, it is a natural extension to consider votes in $\{0, \ldots, N-1\}$ for some integer N. This can be achieved directly by using 1-out-of-N proofs (see [CDS94]). Also, using the above approach for parallel elections, the bits b_i can be assigned weights like 2^{i-1}, and then we get integer votes with $N = 2^K$.

5 Alternative Number Theoretic Assumptions

Under the RSA assumption and the factoring assumption we obtain voting schemes that are comparable in performance and functionality. Note, however, that the initialization of these schemes is more complicated than the initialization of the discrete log based schemes, if it is required that the modulus is generated by multiple parties.

Under the RSA assumption, we have the following scheme. We assume that a modulus N, which is the product of two large distinct primes, a prime v, with $\gcd(v, \phi(N)) = 1$, and an element h of $\mathbb{Z}_N^*$ are available to all parties. The RSA assumption then roughly states that it is infeasible to compute $h^{1/v} \bmod N$. For this setting we have the following efficient commitment scheme. To commit to a value $m \in \mathbb{Z}_v$, the committer chooses $\alpha \in_R \mathbb{Z}_N^*$, and reveals $B = \alpha^v h^m$. The commitment is opened by revealing α and m.

In this setting we may directly apply the technique of [CDS94] to Guillou-Quisquater's identification protocol [GQ88] to construct proofs of validity for ballots of the form $B_i = \alpha_i^v h^{b_i}$. To extend this to a solution with multiple authorities, as in Section 3, we run into a problem as we have no matching verifiably secret sharing scheme. However, an efficient solution for the n-out-of-n case is possible. The idea is simply that the voter chooses shares $a_{ij} \in_R \mathbb{Z}_N^*$, for $j = 1, \ldots, n$, and $b_{ij} \in_R \mathbb{Z}_v$, for $j = 2, \ldots, n$, and sets $\alpha_i = \prod_{j=1}^n a_{ij}$ and $b_{i1} = b_i - \sum_{j=2}^n b_{ij}$. The ballot is then computed as $B_i = \alpha_i^v h^{b_i}$ with $\alpha_i = (\prod_{j=1}^n a_{ij}) h^{(\sum_{j=1}^n b_{ij}) \operatorname{div} v}$. The protocols of Section 3 are translated accordingly.

Similarly for the factoring assumption, these ideas apply to such encryption functions as $\alpha^2 4^b \bmod N$, where N is the product of two primes $p \equiv 3 \pmod 8$ and $q \equiv 7 \pmod 8$.

6 Conclusion

We have presented election schemes of provable security and practical efficiency. The computational and communication complexity are essentially linear, instead of quadratic as for previous schemes. Even with a large number of authorities like $n = 10$, a voter need not communicate more than approximately 10.000 bits of information to submit a vote, assuming that $|p| = 512$ bits and $|q| = 160$ bits for our discrete log scheme. (Note that the actual ballot plus its proof of validity require only 1152 bits.) Compare this, for instance, to [SK94] with in the order of a million bits per vote for the same level of security.

A property of voting schemes, recently identified, that we did not consider in the present paper is that of *non-coercibility* [BT94]. Non-coercibility ensures that no voter will obtain, as a result of an execution of the voting scheme, a receipt that can prove how she voted. The intention of this property is to prevent vote-buying and other tactics of voter persuasion. It is possible (and is part of work in progress) that extensions of our ideas can give a scheme which incorporate non-coercibility while maintaining universal verifiability, privacy, robustness, and efficiency.

References

[Ben87a] J. Benaloh. Cryptographic capsules: A disjunctive primitive for interactive protocols. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 213–222, Berlin, 1987. Springer-Verlag.

[Ben87b] J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, Department of Computer Science Department, New Haven, CT, September 1987.

[BT94] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *Proc. 26th Symposium on Theory of Computing (STOC '94)*, pages 544–553, New York, 1994. A.C.M.

[BY86] J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC '86)*, pages 52–62, New York, 1986. A.C.M.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.

[CF85] J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372–382. IEEE Computer Society, 1985.

[Cha81] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Che94] L. Chen. *Witness Hiding Proofs and Applications.* PhD thesis, Aarhus University, Computer Science Department, Aarhus, Denmark, August 1994.

[CP95] L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology—EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181, Berlin, 1995. Springer-Verlag.

[FS87] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, New York, 1987. Springer-Verlag.

[Gen95] R. Gennaro. Achieving independence efficiently and securely. In *Proc. 14th ACM Symposium on Principles of Distributed Computing (PODC '95)*, New York, 1995. A.C.M.

[GQ88] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology—EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Berlin, 1988. Springer-Verlag.

[Oka93] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Berlin, 1993. Springer-Verlag.

[Ped92] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Berlin, 1992. Springer-Verlag.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[SK94] K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 411–424, Berlin, 1994. Springer-Verlag.

[SK95] K. Sako and J. Kilian. Receipt-free mix-type voting scheme—a practical solution to the implementation of a voting booth. In *Advances in Cryptology—EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393–403, Berlin, 1995. Springer-Verlag.

This article was processed using the LaTeX macro package with LLNCS style