A Secure and Optimally Efficient Multi-Authority Election Scheme (1)

RONALD CRAMER
Inst. for Theoretical Comp. Sc., ETH-Z, CH-8092 Zurich - Switzerland
cramer@inf.ethz.ch

ROSARIO GENNARO
IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 - USA.
rosario@watson.ibm.com

BERRY SCHOENMAKERS
DigiCash, Kruislaan 419, NL-1098 VA Amsterdam - The Netherlands
berry@digicash.com

Abstract. In this paper we present a new multi-authority secret-ballot election scheme that guarantees privacy, universal verifiability, and robustness. It is the first scheme for which the performance is optimal in the sense that time and communication complexity is minimal both for the individual voters and the authorities. An interesting property of the scheme is that the time and communication complexity for the voter is independent of the number of authorities. A voter simply posts a single encrypted message accompanied by a compact proof that it contains a valid vote. Our result is complementary to the result by Cramer, Franklin, Schoenmakers, and Yung in the sense that in their scheme the work for voters is linear in the number of authorities but can be instantiated to yield information-theoretic privacy, while in our scheme the voter’s effort is independent of the number of authorities but always provides computational privacy-protection. We will also point out that the majority of proposed voting schemes provide computational privacy only (often without even considering the lack of information-theoretic privacy), and that our new scheme is by far superior to those schemes.

1. INTRODUCTION

In the cryptographic literature, electronic voting protocols are known as the prime examples of secure multi-party computations. Many papers have been written on the subject and by now an extensive list of properties and requirements is generally accepted as desirable. We will consider these properties in this paper, among which are privacy, universal verifiability, and various forms of robustness. Recent advancements have also been particularly concerned with the performance aspect. In this paper we will show under which circumstances it is possible to achieve a scheme with optimal performance for large-scale elections, while at the same time keeping the system simple and provably secure.

In considering the performance of elections it is clear that the main consideration should be the effort required of a voter. Indeed, while governments can (and do nowadays) afford a large organizational effort to hold elections, it is mandatory to make the voting protocol as simple and efficient as possible for the voter-who might be participating from home using a PC or a Web TV.

In this paper we present a simple multi-authority election scheme in which the task of the voter is reduced to the bare minimum. Basically, the voter posts a single encrypted message (ballot) accompanied with a proof that it contains a valid vote. For security parameter $k$, the size of the ballot as well as of its proof of validity is $O(k)$ bits. Moreover, due to the homomorphic properties of the encryption method used, the final tally is verifiable to any observer of the election, while due to the use of a matching fault-tolerant threshold decryption technique, the individual votes will remain private and the (benign or malign) failure of authorities can be tolerated.

We work in the model set forth by Benaloh [1, 2, 3], where the active parties are divided into $l$ voters $V_1, ..., V_l$ and $n$ tallying authorities (talliers) $A_1, ..., A_n$. To achieve universal verifiability all parties have access to a so-called bulletin board. A bulletin board is like a broadcast channel with memory to the extent that any party (including passive observers) can see the contents of it, and furthermore that each active participant can post messages by appending the message to her own designated area. No party can erase anything from the bulletin board.

In this model, voters cast their votes by posting ballots to the bulletin board. The ballot does not reveal any information on the vote itself but it is ensured by an accompanying proof that the ballot indeed contains a valid vote and nothing else. Due to a homomorphic

(1) A preliminary version of this paper appears in the Proceedings of EUROCRIPT'97.
property of the ballots, the final tally ("sum" of all votes) can be obtained and verified (by any observer) against the "product" of all submitted ballots. This ensures universal verifiability.

1.1. Properties for elections

Let us state and discuss the properties of elections considered in this paper. For our model, the first property below (eligibility) will follow directly from the fact that we assume the availability of a bulletin board. The other properties, except for the last one, are achieved by means of cryptographic protocols as proposed in this paper.

Eligibility means that only eligible voters can cast a vote, and that each eligible voter can cast a single vote.

Universal Verifiability ensures that any party, including a passive observer, can check that the election is fair, i.e., that the published final tally is consistent with the correctly cast ballots. This property also includes that any party can check whether ballots are correctly cast, and that only invalid ballots are discarded.

Privacy of an individual vote is assured against any reasonably sized coalition of parties (not including the voter herself). That is, unless the number of colluding parties exceeds a certain threshold, different ballots are indistinguishable irrespective of the contained votes. We say that information-theoretic privacy is achieved when the ballots are indistinguishable independent of any cryptographic assumption; otherwise we will say that computational privacy is achieved.

Robustness means that the faulty behaviour (either benign or malicious) of any reasonably sized coalition of participants can be tolerated. In large-scale elections this includes that no coalition of voters of any size can disrupt the election; in other words, any cheating voter can be detected and discarded.

No vote duplication means that it should be impossible to copy another voter's vote (even without knowing what the copied vote is).

No interaction between voters. For a large-scale election it is unreasonable to require that the voters should all interact with each other as part of the voting protocol.

Receipt-freeness (or non-coercibility) is treated in section 1.4.

Although we are emphasizing the application of our scheme to large-scale elections, it is also suitable for small-scale elections such as boardroom elections. In the latter case it is even conceivable that each voter plays the role of tallying authority as well; a PC network will suffice as computing platform.

1.2. Computational versus information-theoretic privacy

By far, the majority of election protocols that support some level of verifiability (either universal or limited to voters, who can check their own vote) merely provide computational protection of the voter's privacy. For example, the schemes presented by Benaloh et al. [1, 2, 3, 4] all rely on the so-called r-th residuosity assumption. Once this assumption is broken (e.g., when the public modulus is factorized), the content of each individual ballot can be decrypted. Similarly, schemes using anonymous channels or mixes [5] usually rely on computational assumptions. By recovering the private keys of the mixes, an adversary is able to "open" all ballots posted to the first mix. For example, the scheme of [6] relies on the difficulty of computing discrete logs, both for the secrecy of the mixes' private keys and for the contents of the ballots.

The extent to which the lack of information-theoretic privacy is harmful may be difficult to estimate. For instance, it is hard to predict what happens if fifty-year old votes of a U.S. president are published—although breaking the encryption methods for the currently widely used security parameters will probably be much more harmful.

Whether democracy, from a cryptographic standpoint it is necessary to determine the limits for computational and information-theoretic privacy. As an aside we note that the mere use of multiple authorities can be considered a condition as well. Indeed, election protocols have been proposed that try to eliminate this condition, e.g., [7], but the methods used still require conditions regarding the channels connecting the participants. Since in our case the bulletin board is implemented from multiple servers anyway, and it is seen as a necessary primitive for achieving universal verifiability, we will not consider eliminating the use of a distributed tallying authority.

In this paper we will see how far one can go if computational privacy is the goal. For computational privacy it suffices to assume a public broadcast channel (bulletin board) as communication model. To make an election scheme information-theoretically secure, it is generally believed that private channels between voters and authorities are required. In section 6.1 we will look into this aspect.

1.3. Our contributions

The main result of this paper is a fair election scheme in which the complexity of the voter's protocol is linear in the security parameter k—hence optimal. This comprises the computational as well as the communication complexity (in bits). The voter needs to communicate only $O(k)$ bits and to perform $O(k)$ modular multiplications (7). Moreover, the dominating factor for the work

(7) Throughout, we will take a modular multiplication of two $O(k)$ sized numbers as our unit of work.
of an authority is \( mk \). Compared to the scheme of [8], we thus achieve a reduction of the work for each participant by a factor of \( n \).

In the new scheme, the voter just sends a particular ElGamal encryption of the vote plus a proof that it indeed contains a valid vote. The proof prevents the voters from casting bogus ballots, and should be such that no information whatsoever leaks about the actual vote contained in a ballot. The crux is to keep this proof \( O(k) \), and here we follow the approach of [8]. We will need a novel application of the technique of [9] for constructing efficient witness hiding protocols. The resulting proof of validity is a little bit more complicated than in [8], but still requires only a few modular exponentiations. A proof of knowledge similar to our proof of validity has been used by Chen and Pedersen to construct efficient group signatures [10].

Unlike previous schemes based on Benaloh’s approach, however, we will achieve robustness w.r.t. faulty authorities without increasing the work for the voter. To this end, we will employ fault-tolerant threshold cryptosystems instead of (verifiable) secret sharing schemes. In our case there will be only one public key for which the matching private key is shared among the authorities using threshold cryptography techniques ([11] for a survey). The voter posts the ballot encrypted with the public key of the authorities. The private key is never reconstructed, and only used implicitly when the authorities cooperate to decrypt the final tally. The correctness of the decryption will be assured, even in the presence of malicious authorities.

Apart from achieving a strong set of properties, three major achievements of our scheme are:

i) The work required of the voter is minimal. Compared to [8] the work is reduced by a multiplicative factor of \( n \). Although \( n \) is usually much smaller than \( k \), this is still a substantial gain in practice. The work for the authorities and observers is reduced accordingly.

ii) The protocol for the voter remains the same even if \( n \) is variable. Usually \( n \) grows with the desired security of the scheme (the more authorities the less potential that an adversary can corrupt, say, half of them). Using our protocol this growth is “transparent” to the user. This property has very nice applications in simplifying the whole election infrastructure. For example it allows the government to distribute the software for the election protocol once and for all (thus reducing all the problems connected with checking the integrity of such software), while the value of \( n \) can be determined \( a \ posteriori \).

iii) As a bonus, the new scheme can easily be extended using techniques for proactive threshold cryptosystems [12] to leave the system (and its keys) in place for a really long time without fearing that the secret key gets compromised (section 6.3).

The main scheme presented in the paper is based on the security of the ElGamal encryption scheme (which is related to the difficulty of the discrete log problem). In section 5 we describe an alternative construction related to the hardness of factoring.

1.4. Coercion

Benaloh and Tuinstra in [4] introduced the notion of a receipt for electronic election schemes. They showed how previous election protocols all suffer from a common defect: a voter can carry away from the protocol a receipt that proves the way he voted. This receipt can then be used for vote buying or for coercion. Several uncoercible schemes have been presented since the introduction of the concept [6, 13, 14]. We discuss this issue and the relevance of our paper in this context in section 6.2.

2. THE BUILDING BLOCKS

2.1. Bulletin board

The communication model required for our election scheme is best viewed as a public broadcast channel with memory, which is called a bulletin board. All communication through the bulletin board is public and can be read by any party (including passive observers). No party can erase any information from the bulletin board, but each active participant can append messages to its own designated section.

To make the latter requirement publicly verifiable, we assume that digital signatures are used to control access to the various sections of the bulletin board. Here we may take advantage of any public-key infrastructure that is already in place. Also note that by postulating that each participant can indeed append messages to its section, it is implicitly assumed that denial-of-service attacks are excluded. This property is realized by designing the bulletin board as a set of replicated servers implementing Byzantine agreement, for instance, such that access is never denied as long as at most a third of the servers is compromised. Reiter’s work on the Rampart system shows that this can be done in a secure and practical way (e.g., [15, 16]).

2.2. ElGamal cryptosystem

Our election scheme relies on the ElGamal cryptosystem [17, 18]. It is well-known that the ElGamal cryptosystem works for any family of groups for which the discrete logarithm is considered intractable.

Our construction works in subgroups \( G_p \) of order \( q \) of \( \mathbb{Z}_p \), where \( p \) and \( q \) are large primes such that \( q | p - 1 \). Other practical families can be obtained from elliptic curves over finite fields.

We will now briefly describe the ElGamal cryptosystem, where the primes \( p \) and \( q \) and at least one generator
g of \( G_q \) are treated as system parameters. These parameters as well as other independent generators introduced in the sequel should be generated jointly by (a designated subset) of the participants. This can be done by letting the participants each run a copy of the same probabilistic algorithm, where the coinflips are generated mutually at random.

The key pair of a receiver in the ElGamal cryptosystem consists of a private key \( s \) (randomly chosen by the receiver) and the corresponding public key \( h = g^s \), which is announced to the participants in the system.

Given a message \( m \in G_q \), encryption proceeds as follows. The sender chooses a random \( r \in \mathbb{Z}_q \) and sends the pair \((x, y) = (g^r, h^m)\) as ciphertext to the receiving party. To decrypt the ciphertext \((x, y)\) the receiver recovers the plaintext as \( m = y/x^r \) using the private key \( s \).

The security of our scheme rests on the semantic security of the ElGamal cryptosystem.

2.3. Robust threshold ElGamal cryptosystem

The object of a threshold scheme for public-key encryption is to share a private key among a set of receivers such that messages can only be decrypted when a substantial set of receivers cooperate. See [11] for a survey. The main protocols of a threshold system are:

i) a key generation protocol to generate the private key jointly by the receivers,

ii) a decryption protocol to jointly decrypt a ciphertext without explicitly reconstructing the private key. For the ElGamal system described above, solutions for both protocols have been described by Pedersen [19, 20], also taking robustness into account.

Key generation As part of the set-up procedure of the election scheme, the authorities will execute a key generation protocol due to Pedersen [19]. The result of the key generation protocol is that each authority \( A_i \) will possess a share \( s_j \in \mathbb{Z}_q \) of a secret \( s \). The authorities are committed to these shares as the values \( h_j = g^{s_j} \) are made public. Furthermore, the shares \( s_j \) are such that the secret \( s \) can be reconstructed from any set \( \Lambda \) of \( t \) shares using appropriate Lagrange coefficients, say:

\[
s = \sum_{j \in \Lambda} s_j \lambda_{j, \Lambda}, \quad \lambda_{j, \Lambda} = \prod_{i \in \Lambda \setminus \{j\}}^l \frac{l}{l-j}
\]

This is exactly as in Shamir's \((t, n)\)-threshold secret sharing scheme [21]. The public key \( h = g^s \) is announced to all participants in the system. Note that no single participant learns the secret \( s \), and that the value of \( s \) is only computationally protected.\(^3\)

\(\)\(^3\) The private channels assumed in Pedersen's key generation protocol may be implemented using public key encryption and the bulletin board. This suffices for computational security.

Decryption To decrypt a ciphertext \((x, y) = (g^a, h^m)\) without reconstructing the secret \( s \), the authorities execute the following protocol:

1) Each authority \( A_i \) broadcasts \( w_i = x^r \) and proves in zero-knowledge that \( \log_x h_j = \log_x w_j \).

2) Let \( \Lambda \) denote any subset of \( t \) authorities who passed the zero-knowledge proof. By raising \( x \) to both sides of eq. (1), it follows that the plaintext can be recovered as

\[
m = y \prod_{j \in \Lambda} w_j^{\lambda_{j, \Lambda}}
\]

Note that step 2 assures that the decryption is correct and successful even if up to \( n - t \) authorities are malicious or fail to execute the protocol. The zero-knowledge proof of step 1 will be described in the next section.

2.4. Proofs of knowledge for equality of discrete logs

In this section we will consider several protocols that show equality of discrete logarithms. Using the same notation as above, we present proofs of knowledge for the relation \( \log_g x = \log_h y \), whereby a prover shows possession of an \( a \in \mathbb{Z}_q \) satisfying \( x = g^a \) and \( y = h^a \). An efficient protocol for this problem is due to Chaum and Pedersen [22], Fig. 1. This protocol is not known to be zero-knowledge or witness hiding. The following result however suffices for our application (see [9] for definitions of the notions involved).

**Lemma 1** The Chaum-Pedersen protocol is a three-move, public coin proof of knowledge for the relation \( \log_x x = \log_h y \). The proof satisfies special soundness, and is special honest-verifier zero-knowledge.

\[
\begin{array}{lcl}
\text{Prover} & | & \text{Verifier} \\
[\langle x, y \rangle = (g^a, h^m)] & | & \langle a, b \rangle \\
\quad w \in \mathbb{Z}_q & | & \langle a, b \rangle \\
\quad (a, b) \rightarrow (g^a, h^m) & | & a \rightarrow c \\
\quad r \leftarrow w + \alpha c & | & c \in \mathbb{Z}_q \\
\end{array}
\]

\[
\begin{array}{lcl}
\quad \text{c} & | & \text{w} \leftarrow r \\
\quad \text{g}^2 & | & \text{a} \leftarrow x \\
\quad \text{w} & | & \text{b} \leftarrow y
\end{array}
\]

Fig. 1 - Proof of knowledge for \( \log_x x = \log_h y \).

**Proof** The protocol inherits its properties from the underlying Schnorr protocol [23]. Special soundness holds because from two accepting conversations with the same first move \((a, b, c, r)\) and \((a, b, c', r')\), \( c \neq c' \), a witness \( w = (r - r')/(c - c') \) can be extracted satisfying \( x = g^w \) and \( y = h^w \). Honest-verifier zero-knowledge holds because, for random \( c \) and \( r \) we have that \((g^w, x^c, \)

\[\text{ETT}\]
\( y, \) instead of \( G, \) our message space will be \( \mathbb{Z}_q \) with addition as group operation for the message space. That is, instead of \( G, \) our message space will be \( \mathbb{Z}_q \) with addition as group operation for the message space.

Notice that the above protocol is zero-knowledge only against the honest verifier, but this suffices for our purpose. Indeed, jumping ahead a little, in order to make our protocols non-interactive, the verifier will be implemented using either a trusted source of random bits (a beacon as in [3, 24] or using the Fiat-Shamir heuristic [25] which requires a hash function. In the latter case security is obtained for the random oracle model.

If so desired, a perfect zero-knowledge protocol secure against any verifier can be obtained by replacing the challenge space \( \mathbb{Z}_q \) with \( [0, 1] \) and repeating the basic protocol \( k \) times. The drawback is that the complexity of the resulting protocol is \( O(k^2) \). This protocol is directly related to the protocol for “simultaneous discrete log” by Chaum, Evertse, and van de Graaf [26]. Both protocols, being public-coin ones, can be made non-interactive using the Fiat-Shamir heuristic. Alternatively, there is a four-move, \( O(k) \) zero-knowledge protocol due to Chaum ([27, 28] for a proof). This protocol is not public-coin though, which means that it requires interaction.

### 2.5. Homomorphic encryption

Homomorphic encryption schemes form an important tool for achieving universally verifiable election schemes. A general definition of the notion is as follows. Let \( \varepsilon \) denote a probabilistic encryption scheme. Let \( M \) be the message space and \( C \) the ciphertext space such that \( M \) is a group under operation \( \otimes \) and \( C \) is a group under operation \( \oplus \). We say that \( \varepsilon \) is a \( (\oplus, \otimes) \)-homomorphic encryption scheme if for any instance \( E \) of the encryption scheme, given \( c_1 = E_{\varepsilon_1}(m_1) \) and \( c_2 = E_{\varepsilon_2}(m_2) \), there exists an \( r \) such that

\[
c_1 \otimes c_2 = E_{\varepsilon_3}(m_1 \otimes m_2)
\]

Homomorphic encryption schemes are important to the construction of election protocols. If one has a \( (\otimes, \oplus) \) scheme, then if \( c_i \) are the encryptions of the single votes, by decrypting \( c = c_1 \otimes \ldots \otimes c_m \), one obtains the tally of the election, without decrypting single votes.

The ElGamal cryptosystem as presented above already satisfies this definition, where the message space is \( G_q \) with multiplication modulo \( p \) as group operation, and the ciphertext space is \( G_q \times G_q \) with componentwise multiplication modulo \( p \) as group operation. Namely, given an ElGamal encryption \((x_1, y_1)\) of \( m_1 \) and an ElGamal encryption \((x_2, y_2)\) of \( m_2 \), we see that \((x_1, x_2, y_1, y_2)\) is an ElGamal encryption of \( m_1 \otimes m_2 \).

For the reasons sketched above however, we need to take this one step further to a homomorphic scheme with addition as group operation for the message space. That is, instead of \( G_q \), our message space will be \( \mathbb{Z}_q \) with addition modulo \( q \) as group operation. Given a fixed generator \( G \in G_q \), the encryption of a message \( m \in \mathbb{Z}_q \) will be the ElGamal encryption of \( G^m \). The observation is now that, given two such encryptions of \( m_1 \) and \( m_2 \), respectively, the product is an encryption of \( m_1 + m_2 \) modulo \( q \). Notice that for such a scheme decryption involves the computation of a discrete log, which is a hard task in general. Nevertheless it can be done efficiently for “small” messages, as will be the case in our election scheme (section 3).

#### 2.6. Efficient proofs of validity

In our election each voter will post an ElGamal encryption of either \( m_0 \) or \( m_1 \), where \( m_0 \) and \( m_1 \) denote distinct elements of \( G_q \). Later we will consider suitable values for \( m_0 \) and \( m_1 \). The encryption should be accompanied by a proof of validity that proves that the encryption indeed contains one of these values. Furthermore, the proof should not reveal any information about which one.

Consider an ElGamal encryption of the following form:

\[(x, y) = (g^a, h^a \cdot m) \quad \text{with} \quad m \in \{m_0, m_1\}\]

where the prover knows the value of \( m \). To show that the pair \((x, y)\) is indeed of this form without revealing the value of \( m \) boils down to a witness indistinguishable proof of knowledge of the relation given by:

\[
\log_g x = \log_g (y/m_0) \quad \lor \quad \log_g x = \log_g (y/m_1)
\]

This protocol is directly related to the protocol for “simultaneous discrete log” by Chaum, Evertse, and van de Graaf [26]. Both protocols, being public-coin ones, can be made non-interactive using the Fiat-Shamir heuristic. Alternatively, there is a four-move, \( O(k) \) zero-knowledge protocol due to Chaum ([27, 28] for a proof). This protocol is not public-coin though, which means that it requires interaction.

<table>
<thead>
<tr>
<th>Voter</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v = 1 )</td>
<td>( v = -1 )</td>
</tr>
<tr>
<td>( \alpha, w, r_1, d_1 \in \mathbb{Z}_q )</td>
<td>( \alpha, w, r_1, d_1 \in \mathbb{Z}_q )</td>
</tr>
<tr>
<td>( x \leftarrow g^a )</td>
<td>( x \leftarrow g^a )</td>
</tr>
<tr>
<td>( y \leftarrow h^a \cdot G )</td>
<td>( y \leftarrow h^a \cdot G )</td>
</tr>
<tr>
<td>( a_1 \leftarrow g^x \cdot x )</td>
<td>( a_1 \leftarrow g^x \cdot x )</td>
</tr>
<tr>
<td>( a_2 \leftarrow g^y \cdot y )</td>
<td>( a_2 \leftarrow g^y \cdot y )</td>
</tr>
<tr>
<td>( b_1 \leftarrow h \cdot (yG)^{a_2} )</td>
<td>( b_1 \leftarrow h \cdot (yG)^{a_2} )</td>
</tr>
<tr>
<td>( b_2 \leftarrow h \cdot (yG)^{a_2} )</td>
<td>( b_2 \leftarrow h \cdot (yG)^{a_2} )</td>
</tr>
<tr>
<td>( d_2 \leftarrow c - d_1 )</td>
<td>( d_2 \leftarrow c - d_1 )</td>
</tr>
<tr>
<td>( r_2 \leftarrow w - \alpha d_2 )</td>
<td>( r_2 \leftarrow w - \alpha d_2 )</td>
</tr>
</tbody>
</table>

\( c \leftarrow \frac{d_2 + c_2}{d_2 + d_2} \)

\( a_1 \leftarrow g^x \cdot x \)

\( b_1 \leftarrow h \cdot (yG)^{a_1} \)

\( a_2 \leftarrow g^y \cdot y \)

\( b_2 \leftarrow h \cdot (yG)^{a_2} \)

Fig. 2 - Encryption and Proof of Validity of Ballot \((x, y)\).
have prepared Lemma 1. On account of this lemma, we have that the protocol exactly satisfies the conditions for the construction of [9]. Fig. 2 for a preview of the protocol, as it is used in the election scheme of the next section.

3. Multi-authority election scheme

Given the primitives of the previous section we now assemble a simple and efficient election scheme. The participants in the election protocol are n authorities A1,...,An and / voters V1,...,V/. Recall that the requirements for a ballot are that it must contain a vote in an unambiguous way such that:

i) votes accumulate when ballots are aggregated,
ii) the proof of validity shows that a ballot contains either a yes-vote or a no-vote, without revealing any information on which of the two is the case.

To show that the same masking technique as in [8, 29] can be used, we instantiate the scheme of section 2.6 with m1 = G and m2 = 1/G, where G is a fixed generator of G. Thus a ballot is prepared as an ElGamal encryption of the form (x, y) = (g^b, h^b G) for random b ∈ R {1, -1}, and the corresponding proof of knowledge is depicted in Fig. 2.

In order to make vote casting non-interactive two approaches can be taken. One is to assume the presence of a trusted beacon that broadcasts random bits at regular intervals. In this case the challenge c can be taken from the bits sent by the beacon when the voter posts her vote (assuming that there is a mechanism that assigns unique slots to voters). A practical way to implement a beacon is the standard use of a suitable hash function H to compute the challenge c from the first message of the proof. In this case security is retained in the random oracle model, but some care is required to prevent vote duplication. Each challenge must be made voter-specific [30], i.e., the challenge c is computed by voter Vi as H (IDi, x, y, a1, b1, a2, b2), where IDi is a unique public string identifying Vi.

To cast a ballot the voter posts an additional number e ∈ {1, -1} such that v = be is equal to the desired vote. Alternatively, voters may adapt the precomputed values before sending the ballot out, i.e., precompute (x, y) and then post (x', y').

As part of the initialization the designated parties generate the system parameters p, q, g, G, as described in section 2.2, where we may safely assume that l < q/2 for any reasonable security parameter k. Secondly, the authorities execute the robust key generation protocol as described in section 2.3. The transcripts of these protocol should appear on the bulletin board. Note that this also shows to any observer that indeed n authorities are taken part in the scheme, which is otherwise not visible to the voters (4).

The main steps of the voting protocol now are, where we assume w.l.o.g. that only correct ballots are cast:

1) Voter Vi posts a ballot (xi, yi) to the bulletin board accompanied by a non-interactive proof of validity.
2) When the deadline is reached, the proofs of validity are checked by the authorities and the product (X, Y) = (Πi=1 xi, Πi=1 yi) is formed.
3) Finally, the authorities jointly execute the decryption protocol of section 2.3 for (X, Y) to obtain the value of W = Y/X. A non-interactive proof of knowledge is used in Step 1 of the decryption protocol.

We thus get W = G^T as a result, where T is equal to the difference between the number of yes-votes and no-votes, -l ≤ T ≤ /l. Hence, T = logGW which is in general hard to compute. However, in our case we can now fully exploit the fact that the number of voters / is relatively small—certainly polynomial in the security parameter! The value of T can be determined easily using O(/) modular multiplications only, by iteratively generating G^i, G^{t+1}, G^{2i},..., (each time using one multiplication) until W is found. Asymptotically, the work does therefore not increase for the authorities (at most two multiplications per voter). Note also that the computation of logGW may be done by any party because the result is verifiable (5).

The time and communication complexity of the scheme is as follows. The work for a voter is clearly linear in l, independent of the number of authorities. The work for the authorities is only O((k + n)k) (assuming that the zero-knowledge proof used in step 3 is O(k), hence negligible). Since we may safely assume that the number of voters is larger than the number of authorities, the work for the authorities is actually O(lk).

Similarly, the work for an observer who wants to check the outcome of the election is O(lk).

We summarize our result in the following theorem.

Theorem 1 If the ElGamal cryptosystem is semantically secure, then our election scheme provides universal verifiability, computational privacy, robustness, and prevents vote duplication.

Proof Universal verifiability is achieved because any observer can check the proofs of validity for the ballots, since these are made non-interactive. It is also clear to any observer if the final tally is correct with respect to all valid ballots.

Privacy of individual votes is guaranteed by the

---

4) This is because messages to the bulletin board are authenticated using digital signatures, so it is not possible that a single entity simulates the behavior of a authorities whose public keys are commonly known to the observers.

5) If this O(/) search method is considered too slow for a large-scale election, Shanks' baby-step giant-step algorithm (e.g., [31, section 3.1]) can be applied to find T in O(/l) time using O(/l) bits of storage.
security of the ElGamal cryptosystem used to encrypt the votes. This is true because we assume that no more than \( t - 1 \) authorities conspire, since \( t \) authorities can reconstruct the secret key used in the scheme. Besides the encrypted votes a voter casts a proof of validity, but this is useless in order to break the privacy since such proof is witness indistinguishable.

Robustness with respect to malicious voters is achieved by means of the soundness of the proof of validity, which ensures that voters cannot submit bogus ballots. Robustness with respect to at most \( n - t \) malicious authorities is inherited from the robustness of the key generation and decryption protocols.

Finally, vote duplication is prevented due to the fact that the proofs of validity are made voter-specific.

For the non-interactive version of the scheme based on the Fiat-Shamir heuristic, the result holds in the random oracle model.

### 4. Extension to Multi-way elections

Instead of offering a choice between two options, it is often required that a choice between several options can be made. There are numerous approaches to tackle this problem. Below, we sketch an approach for which the computation of a different set of \( t \) choices, we simply take \( K \) (independently generated) one generator for each option separately. The proof of validity of a ballot (\( y \)) now boils down to a proof of knowledge of \( \log_y x = \log_y (y/G_i) \lor \ldots \lor \log_y x = \log_y (y/G_K) \)

Since the voter can only generate this proof for at most one generator \( G_i \), it is automatically guaranteed that the voter cannot vote for more than one option at a time.

The problem of computing the final tally is in general more complicated. After decryption by the authorities, a number \( W \) is obtained that represents the final tally, \( W = G_1^{T_1} \ldots G_K^{T_K} \), where the Ti's form the result of the election. Note that the Ti's are uniquely determined by \( W \) in the sense that computation of a different set \( T_i' \)'s satisfying \( W = G_1^{T_1'} \ldots G_K^{T_K'} \) would contradict the discrete log assumption, using the fact that the generators \( G_i \) are independently generated. Since \( T_i \geq 0 \) and \( \sum_{i=1}^{k} T_i = l \), computation of the Ti's is feasible for reasonable values of \( l \) and \( K \).

5. Alternative number-theoretic assumption

To show the generality of our approach we now present a scheme for which the security is related to the difficulty of factoring. Specifically, we present a scheme based on the \( q \)-th residuosity assumption (as in the original Benaloh schemes). The notion of \( q \)-th residues is an extension of quadratic residues. A number \( x \) is a \( q \)-th residue modulo \( N \) if there exists an \( \alpha \) such that \( \alpha^q = x \pmod{N} \). It is believed to be hard to distinguish between \( q \)-residues and non \( q \)-residues.

This suggests the following homomorphic encryption scheme. We present a specific implementation which is suitable to threshold cryptography techniques. The parameters of the scheme are a modulus \( N = P Q \), where \( P = 2P' + 1 \) and \( Q = 2Q' + 1 \), with \( P, Q, P', Q' \) all large primes. As before, the prime \( q \) can thus be assumed to be larger than twice the number of voters \( l \). The public key must include a fixed number \( Y \in \mathbb{Z}_N^* \), which is not a \( q \)-th residue modulo \( N \).

We will consider messages from \( \mathbb{Z}_N^* \). The ciphertext for a message \( m \) is now \( E_m(m) = \alpha Y^m \), where \( \alpha \in \mathbb{Z}_N^* \). As before, decryption is hard, in general, but in our case an exhaustive search for all possible values suffices. The right \( m \) is detected when by computing \( (cY^{-m})^{QN/\phi(N)} \pmod{N} \) one gets back \( 1 \). Note that \( c' = c^{QN/\phi(N)} \pmod{N} \) and \( Y' = (Y^{-1})^{QN/\phi(N)} \pmod{N} \) can be computed first, and then test for \( c' Y^m \), where \( m \) is selected from all possible messages.

Next we discuss a robust threshold cryptosystem for this setting. Notice that the value \( d = \phi(N)/q \) could be considered the secret key of the scheme, and that decryption is carried out by simply computing exponentiations (modulo \( N \)) with exponent \( d \). As the setting is very similar to an RSA decryption, we can apply the result of [32] to obtain an efficient and robust threshold decryption procedure. The result in [32] holds for RSA moduli which are the product of safe primes (i.e., \( P = 2P' + 1 \) and \( Q = 2Q' + 1 \)), but it can easily be extended to work for our specific needs.

The key generation protocol, however, relies on secure multiparty computations as there is no known efficient way to perform a distributed key generation algorithm for factoring based schemes. However, since this task is part of the set-up of the scheme, this may be acceptable as a one-time operation.

Our final task is to construct an efficient proof of validity that shows that a ballot \( x \) is correctly formed. This amounts to showing that \( x = \alpha Y^v \), for some \( \alpha \), with \( v \in \{-1, 1\} \), hence that either \( x/Y \) or \( xY \) is a \( q \)-th residue. As before, Lemma 2 below guarantees the existence of an efficient proof of validity, based on the construction of [9].

**Lemma 2** The protocol of Fig. 3 is a three-move, public coin proof of knowledge for \( r \)-th residuosity. The proof satisfies special soundness, and is special honest-verifier zero-knowledge.
Proof: Completeness of the proof of knowledge is clear. Special soundness holds because for two accepting conversations \((a, c, r)\) and \((a', c', r')\), where \(b\), it follows that \((r/r')y = x^{-c}c'\). Since \(0 < c - c' < q\) we have that there exist integers \(k, l\) such that \((c - c')k = 1 + lq\), hence \((r/r')y = x^{q+1}\), which may be rewritten as \([r/r']y x^{q+1} = x\). Honest-verifier zero-knowledge holds because, for random \(c \in \mathbb{Z}_q^*\) and \(r \in \mathbb{Z}_q^*\) we have that \((r/r')y x^{q+1} = x\). An accepting conversation with the right distribution. Since challenge \(c\) can be chosen freely, we also have special honest-verifier zero-knowledge.

\[\square\]

**Theorem 2** Under the \(q\)-th residuosity assumption, our election scheme provides universal verifiability, computational privacy, robustness, and prevents vote duplication.

![Proof](proof.png)

**Fig. 3 - Proof that \(x\) is an \(q\)-th residue.**

6. **Discussion**

6.1. **Information-theoretically secure elections**

The scheme of [8] in principle provides information-theoretic protection of the voter's privacy. This is due to the fact that voters post (a number of) information-theoretically hiding commitments to the bulletin board and that these commitments are opened to the authorities using private channels. A general problem with such a solution is that the use of private channels opens the possibility for disputes: on the one hand a dishonest voter may just skip sending a message to an authority, while on the other hand a dishonest authority may claim not to have received a message.

It is therefore worthwhile to limit the possibility for disputes to the set-up process for the election. During the election protocol itself no disputes on the usage of the private channel should be possible. The idea is to use a public broadcast channel (such as a bulletin board) on which the parties post commitments to mutually selected keys. Each pair of parties first agrees on a key using a secure channel. Only if both parties broadcast the same commitment, the set-up of the private channel succeeded. Otherwise, there is dispute that must be solved at this stage. It is important that:

i) the commitment is information-theoretically hiding,

ii) the encryption method is information-theoretically secure (a one-time pad). More concretely, the two phases are as follows.

**Set-up** Both parties agree on a mutually at random selected key \(K\) and a commitment \(B\) on this key. Both parties broadcast a signed copy of the commitment. The key set-up is only successful if both parties broadcast the same commitment. Disputes in this stage have to be resolved in a procedural way.

**Communication** To send a message \(m\), the sender will broadcast the encryption \(E_K(m)\) over the public channel. Only the intended receiver is able to recover the message.

Using this method, private channels can be set up from each voter \(V_i\) to each authority \(A_j\). Once set up succeeds there can be no dispute on the use of the private channel. Anybody sees if the voter abstains from posting the required values to the bulletin board. If what the voter submits consists of incorrect shares, the respective authorities open the commitments to the key so that this fact can be verified. Note that for the scheme of [8] the use of the private channels is limited to two elements of \(\mathbb{Z}_q^*\) per channel.

6.2. **Incoercible protocols**

Receipt-free or incoercible election scheme that have been proposed so far all rely on some form of physical assumption [4, 6, 13]. The minimal assumption required (as in [6]) is the existence of a private channel between the voters and the authorities. These schemes allow a voter to lie about the vote cast even if under coercion, but not up to the level that coercer who exactly prescribe which private random bits the voter must use can be withstood. Indeed given the execution of the protocol the voter will be able to create two different histories of his computations, both consistent with the execution but corresponding to two different votes. All these schemes also require that the authorities are incoercible, or alternatively that voters know which ones have been coerced. Moreover, as pointed out in the previous section, the use of private channels gives rise to disputes. (Another viable approach is to assume that the voters dispose of a tamper-proof encryption box such as a smartcard, but we consider this beyond the scope of this paper).

Recently, Canetti and Gennaro in [14] proved that general secure multiparty computation protocols can be made incoercible without the above assumptions, in particular without assuming untappable channels. Their scheme is based on a new type of encryption called \(\text{deniable encryption}\) introduced in [33] that allows a sender to encrypt a bit \(b\) in such a way that the resulting ciphertext can be "explained" as either \(b\) or \(1-b\) to a coercer. The construction in [14] works for the general problem of secure multi-party computation; as such it is described in terms of a complete network of communi-
e\n
cation and the result holds as long as at most half of the
players in the network are coerced. For the case of elec-
tion schemes, the construction of [14] can be scaled
down to the bulletin board model (thus not requiring
communication between voters). In this model all voters
can withstand coercion provided the coercer is not able
to prescribe the random bits of the voters, and at most
half of the authorities can be completely coerced. The
complexity of the resulting scheme is high (although
polynomial), but opens the door to the search for effi-
cient incoercible schemes.

In order to make our election scheme incoercible
(without physical assumptions) we would need a deni-
able encryption scheme which is

i) homomorphic, 

ii) suitable to threshold cryptography techniques.

An interesting open problem is thus to construct
such a scheme.

6.3. Proactive security

The secrecy of the votes is protected against coal-
tions of up to \( t - 1 \) authorities. In other words, an attack-
er must recover \( t \) shares of the private key in order to be
able to decrypt single votes. This is similar to previous
protocols in which the vote is \((t, n)\)-shared among the
authorities. We note that the use of threshold cryptogra-
phy instead of secret sharing presents also some advan-
tages in this area. Using proactive security techniques
[12, 34, 35] it is possible to leave the public key of the
system in place for a really long time without fearing it
being compromised. Indeed, when using proactive
schemes the shares of the private key are periodically
"refreshed" so that an attacker is forced to recover \( t \)
shares in one single period of time that can be as short
as a day. Both schemes presented in this paper can be
made proactive, the discrete-log based one using the
techniques in [12] and the factoring one by adapting
the work of [35].

Let us briefly describe how this stronger security
aspect can be achieved. For simplicity we will refer just
to the discrete log based scheme [12]. At each “refresh-
ning” deadline (say every day at midnight), the autho-
rities run the key generation protocol, but now sharing a
zero value. The new shares are added to the old shares
of the secret key \( s \). The resulting shares still interpola-
te to \( s \) (since the free term of the polynomial is unchanged)
but lie on an otherwise different polynomial. This
implies that old shares are now useless if combined with
new ones, and therefore the attacker needs to start col-
clecting shares from scratch.

7. Conclusion

We have shown a very efficient scheme for secure
elections based on the discrete log assumption, and a
somewhat more complicated scheme based on the \( q \)-th
residuosity assumption. The new schemes satisfy all
well-known requirements, except for receipt-freeness.
An open problem is to construct efficient incoercible
election protocols, preferably without relying on physical
assumptions.

In our scheme the work for the voter is minimal and
independent of the number of authorities. Election
 schemes based on the mix channel of [36] also have this
property but for several reasons our approach is prefer-
able over those schemes. In mix-based schemes the
final tally is computed by somehow decrypting the indi-
vidual ballots, while in our approach a single decryption
of the aggregate of the ballots suffices. In mix-based
schemes disrupters may submit invalid ballots which
are detected only after decryption has taken place; in
our scheme disruption by voters is automatically pre-
vented because of the required proof of validity for bal-
lots. Another important difference is that due to the use
of a threshold cryptosystem we achieve robustness in a
stronger sense. Indeed in mix-based schemes the failure
of a single authority would compromise the whole pro-
tocol. In our case we can tolerate malicious behavior of
a constant fraction (half) of authorities. Finally, the
security of our scheme can be proven from its construc-
tion, while some security problems with the schemes of
[6, 36] exist, as shown for instance in [37, 38].

We would like to emphasize that the work for the
voter is really low. For example, for the discrete log
scheme, we have for \(| p | = 64 \) bytes and \(| q | = 20 \)
bytes, that the size of the ballot plus its proof plus a sig-
nature on it is only 272 bytes in total. Clearly, this is an
order of magnitude better than [8], which was already
two orders of magnitude better than any previous
scheme. Furthermore, computation of the ballot and its
proof require a few exponentiations only Fig. 2. A
direct consequence of the reduced ballot size is also that
the task of verifying the final tally is much simpler.

Manuscript received on November 15, 1996.

REFERENCES

secure election scheme. In Proc 26-th IEEE Symposium on
Foundations of Computer Science (FOCS '85). IEEE Computer

enhance the privacy of voters. In Proc. 5-th ACM Symposium
on Principles of Distributed Computing (PODC -86). New

University, Department of Computer Science Department,

Proc. 36-th Symposium on Theory of Computing (STOC '94),

[5] D. Chaum: Untraceable electronic mail, return addresses, and


