A Zero Knowledge proof for Subset Selection from a Family of Sets with applications to Multiparty/Multicandidate Electronic Elections

Tassos Dimitriou[1] and Dimitris Foteinakis[2]

[1] Athens Information Technology
Markopoulo Ave., 19002, Athens, Greece
tdim@ait.gr
http://www.ait.gr
[2] Intracom S.A.
Markopoulo Ave., 19002, Athens, Greece
fotd@intracom.gr
http://www.intracom.gr

Abstract. We present a methodology for proving in Zero Knowledge the validity of selecting a subset of a set belonging to predefined family of sets. We apply this methodology in electronic voting to provide for extended ballot options. Our proposed voting scheme supports multiple parties and the selection of a number of candidates from *one and only one* of these parties. We have implemented this system and provide measures of its computational and communication complexity. We prove that the complexity is linear with respect to the total number of candidates and the number of parties participating in the election.

1 Introduction

With the explosion of growth of the World Wide Web, the increase of computational power, computer memory and the storage capacity, we have the ability to communicate more information faster, cheaper and more reliably. The general trend towards a paperless society has affected the area of voting. Many attempts have been made to create systems that would allow modern computer-based technology to emulate the secure desirable properties valued in centuries of public voting.

Remote electronic voting refers to an election process whereby people can cast their votes over the Internet, from the comfort of their home, or possibly any other location where they can get Internet access. There are many aspects of elections besides security that bring this type of voting into question. The primary ones are:

- *Coercibility*: The danger that outside of a public polling place, a voter could be coerced into voting for a particular candidate.
- *Vote selling*: The opportunity for voters to sell their vote.

- *Vote solicitation:* The danger that outside of a public polling place, it is much more difficult to control vote solicitation by political parties at the time of voting.
- *Invalid Registration:* The issue of whether or not to allow online registration, and if so, how to control the level of fraud.

The possibility of widely distributed locations where votes can be cast changes many aspects of our carefully controlled elections as we know them. The relevant issues are of great importance and could very well influence whether or not such election processes are desirable. However, in this paper we do not discuss issues like the vulnerability of the Internet to denial of service attacks, the unreliability of the Domain Name Service or the various threats the supporting hosts are liable to, but we focus instead on the security considerations of the *voting process*.

Thanks to the advances in the fields of cryptography an electronic voting system can satisfy the requirements that are considered self-evident for paper based systems and at the same time be efficient and reduce the cost of large scale elections. In [10] Fujioka *et al.* define the properties of a secure, secret election:

- *Completeness*: All valid votes must be counted correctly.
- *Soundness*: The dishonest voter cannot disrupt the voting process.
- *Privacy* and *Integrity*: All votes must be and remain secret and cannot be altered in transit. Hence effective encryption must be used to protect the votes from being disclosed to third parties during transmission.
- *Anonymity*: The voting system must support the voters' right to secrecy of their vote. Hence the vote recording mechanism must not identify the individual voter.
- *Unreusability* and *Eligibility*: No voter can vote twice and no one who isn't allowed to vote can vote.
- *Fairness*: Nothing must affect the voting process. Voters must not be able to affect the system if the supply invalid ballots, colluding or fault authorities must not alter the voting process's results.
- *Verifiability*: Any external party can verify that the result of the election is correct. In particular, this means that votes are recorded as captured and cannot be manipulated when transferred from vote collection to tabulation.

Many voting schemes have been proposed so far that satisfy the above set of requirements, but they are limited by the ballot options they can support. Initial voting systems [1] allowed the voter to select one of two candidates ("yes/no" paradigm) and later systems can accommodate the selection of "t out of n" candidates [3], [4], [6], [8]. Such electoral systems are encountered in the United States and electronic voting systems have already been applied in various State elections. These systems however, cannot be used for elections whose structure is more complicated than a selection of "t out of n" candidates like many electoral systems throughout Europe.

In this work we will present an election system that will accommodate *multiple parties* which in turn consist of *multiple candidates* and will allow a voter to select "t out of n" candidates within a *single party* and prove in zero knowledge manner that this selection is valid.

2 Related Work

An electronic voting scheme consists of a set of protocols which allow voters to cast ballots while a group of authorities collect the votes and output the final tally. Election schemes in [1], [2], [3], [4] were first described by Benaloh [1]. All these schemes mainly discuss the "yes/no" vote scenario. Such schemes utilize a cryptosystem that has the *homomorphic* property (Section 4.1) which allows the computation of the tally without the decryption of individual ballots. Two different election models have been proposed so far.

In Benaloh schemes, a voter shares his vote between n authorities so that t of them can recover it. Each authority computes its encrypted share of the tally and finally t of them need to collaborate to compute the actual tally.

In schemes like [4], the voters send their encrypted votes to a single *combiner*. Using the homomorphic property of the cryptosystems the combiner computes the encrypted tally in a public verifiable way. Then t authorities need to collaborate in order to recover the tally by running a threshold cryptosystem.

Systems described in [3], [4], which are implemented in commercial electronic voting applications, use a variant of the El Gamal cryptosystem exhibiting the homomorphic property and having a threshold version [11]. This cryptosystem requires exponent search in order to compute the final tally from the ballot product, an operation that is computationally *expensive* and may render the system inefficient when the number of voters increases.

Our system uses the Paillier Cryptosystem [5] which exhibits the homomorphic properties needed by the election scheme and more importantly provides an efficient decryption algorithm as well as the largest bandwidth among all cryptosystems using a trapdoor to compute the discrete logarithm. A threshold version of the cryptosystem that appeared in [6], [7] allows its usage in the case of multiple authorities to jointly run a threshold cryptosystem and collaborate in order to decrypt the final tally.

3 Our Contribution

In this paper we present a methodology for proving in a Zero Knowledge manner the validity of *selecting a subset of a set belonging to a predefined family of sets*. More specifically we show how to prove in Zero Knowledge that the voter's selection consists of t elements (candidates) all belonging to the *same* set (party) out of k sets of at most n elements each. We apply this methodology in order to construct an election scheme that will accommodate multiple parties and the selection of t candidates from *one and only one* of those parties. Such election systems are common throughout the world, like in many European countries. Electoral systems with the above mentioned structure are referred to "open party list" systems and are used in many countries, among which are the Netherlands, Norway, Greece, Spain and Slovenia. Electronic voting schemes proposed up to now were limited to the selection of "t out of n" candidates and could not accommodate such a complex system. Our proposal extends existing voting schemes and can be applied to elections which have those requirements.

Furthermore we have implemented a voting system that uses our proposed protocol in order to run an election. Our scheme guarantees the necessary requirements of a voting system mentioned in the introduction: privacy of voters, public verifiability, fairness and soundness, eligibility and unreusability. *Privacy of voters* guarantees that a vote will be kept secret from any collusion of t or less authorities, where t is a system parameter. *Public verifiability* ensures that any external party can verify that the election is fair and that the published tally was computed correctly from the ballots that were correctly cast, through the bulletin board. *Fairness* and *Soundness* ensure that the system can tolerate up to t faulty or colluding authorities without its operation being affected. The existence of the bulletin board and of secure channels via public key cryptography ensures *eligibility* and *unreusability*.

Finally we performed a set of experiments to measure the time taken for creation and verification of the Zero Knowledge proofs, the size of produced ballots as well as the total time needed for a voter to cast a vote in our system, including user and server authentication, ballot creation, transmission and verification, for various system parameters. We prove and validate experimentally that the running time of the algorithm for the creation of the ballot and the Zero Knowledge proofs is linear with respect to the total number of candidates in the elections. Hence our system not only fulfills the basic security requirements but it is also user friendly and practical, an essential characteristic of any voting system that is to be used for public elections.

The outline of the paper is as follows: The next section describes the cryptographic tools that our system uses. We briefly describe the Paillier Cryptosystem and its properties as well as two Zero Knowledge proofs that are used as building blocks for the creation of more complex ones. Section 5 describes the voting protocol, the ballot creation procedure and the Zero Knowledge proofs creation. In Section 6 we describe how the voting system is implemented, the steps needed to cast a ballot as well as the steps for calculating the final result. Furthermore we provide a complexity analysis of the proposed system and measurements of its operation. Finally, we conclude in Section 7.

4 Cryptographic Tools

4.1 The Paillier Cryptosystem

In [5], Paillier proposes a new probabilistic encryption scheme based on computations in the group $\mathcal{Z}^*_{\mathcal{N}^2}$, where $\mathcal{N}$ is an RSA modulus. This scheme has some very attractive properties: It is homomorphic, allows encryption of many bits in one operation with a constant expansion factor and allows efficient decryption. This cryptosystem is based on the *Decisional Composite Residuosity Assumption* (DCRA).

Description of Paillier Scheme

- *Key Generation*: Let $\mathcal{N}$ be an RSA modulus $\mathcal{N} = pq$, where p and q are prime integers. Let g be an integer of order a multiple of $\mathcal{N}$ modulo $\mathcal{N}^2$. The public key is $PK = (N, g)$ and the secret key is $SK = \lambda(\mathcal{N})$ where $\lambda(\mathcal{N})$ is defined as $\lambda(\mathcal{N}) = lcm[(p-1)(q-1)]$. We should note that in [6] Damgraad and Jurik propose that $g = N + 1$ can be used without degrading security. Then the public key will only consist of $\mathcal{N}$.
- *Encryption*: To encrypt a message $M \in \mathcal{Z}_{\mathcal{N}}$ choose a random $r \in \mathcal{Z}^*_{\mathcal{N}}$ and compute the ciphertext $c = g^M r^N \mod \mathcal{N}^2$.
- *Decryption*: To decrypt a ciphertext c, compute:

$$M = \frac{L(c^{\lambda(\mathcal{N})} \mod \mathcal{N}^2)}{L(g^{\lambda(\mathcal{N})} \mod \mathcal{N}^2)} \mod \mathcal{N}^2, \tag{1}$$

where the L-function receives input from the set $S_{\mathcal{N}} = \{u < \mathcal{N}^2 | u = 1 \mod \mathcal{N}\}$ and

$$L(u) = \frac{u-1}{\mathcal{N}} \ . \tag{2}$$

Homomorphic Property

The encryption function has an "algebraic property" which allows computations with the encrypted values without knowing the contents of the ciphertexts. More precisely the encryption function has the following property: $E(M_1 + M_2) = E(M_1) \cdot E(M_2)$ and consequently $E(k \cdot M) = E(M)^k$.

This property is necessary to achieve anonymity as well as universal verifiability since the tally can be computed without the decryption of individual votes by all interested parties. The decryption is performed on the final tally, guaranteeing the privacy of the voters.

In [3] as well as in other voting schemes a variant of the El Gamal encryption scheme is used. In this variant in order to encrypt a message m we compute ($g^k \mod p$ and $g^m y^k \mod p$) instead of ($g^k \mod p$ and $my^k \mod p$). In this manner the scheme gains the homomorphic property needed for the election schemes. The negative side of this encryption scheme is that no trapdoor exists to compute m given $g^m \mod p$. Therefore this scheme is only used in "yes/no" schemes, where the message m is either 0 or 1 and therefore g^m lies in limited subset of messages and can be computed using exhaustive search. However, this scheme has been used in several voting systems including some commercial ones. In the case of elections with many candidates and a large number of voters the modified El Gamal scheme becomes inapplicable since exhaustive search or more efficient methods like index calculus cannot efficiently compute the tally.

Threshold version of Paillier Cryptosystem

A (t, n) threshold scheme does not reveal a secret S unless any t out of the n participants work together. In order to prevent authorities to learn the contents of the votes submitted and to ensure the privacy of the voters a threshold version

of the Paillier Cryptosystem can be used. In this case, instead of having a single authority decrypt the encrypted tally, n authorities share the secret, so that at least t are needed to perform the decryption operation. Such versions of the cryptosystem are presented in [6] and [7]. A general description of a threshold decryption model follows:

The scheme includes the following participants: a combiner, a set of n authorities A_i and users.

- In the initialization phase, the authorities run a distributed key generation protocol to create the public key PK and the secret shares SK_i of the private key SK with or without a trusted dealer [12], [13]. Next the authorities publish the verification keys VK, VK_i.
- The user encrypts a message using the private key PK.
- To decrypt a ciphertext c, the combiner forwards c to all the authorities. Using the shares of the secret key SK_i and the verification keys VK and VK_i each authority runs the decryption algorithm and produces a partial decryption c_i, providing a proof of validity for the partial decryption. The combiner can then produce the decryption of ciphertext c, if enough partial decryptions (t or more) are valid.

4.2 Zero Knowledge Proofs

Central to our results is the way to achieve an efficient proof of validity for ballots. The proof of validity shows to any interested party including the tabulation authorities that a ballot actually represents a valid vote. To maintain the privacy of the voters this proof of validity will be a zero knowledge proof. In general, a Zero Knowledge proof allows an all powerful prover to convince a verifier about the validity of statement without leaking any information other than its correctness.

The efficiency of the entire voting scheme depends greatly on the efficiency of the zero knowledge proofs in terms of computational effort and in terms of the required bandwidth. Our goal is to create a zero knowledge proof that the submitted ballot is a selection of t candidates from a *single* party. Since this a complex proof we will use as building blocks the zero knowledge proofs presented in [6] which we include here for completeness. We note that the following protocols are not zero knowledge as they stand; only honest verifier zero knowledge. However, zero knowledge protocols for the same problems can be constructed using standard methods and secondly in our applications we will always use them in a non interactive variant based on the Fiat-Shamir heuristic [9].

In the following protocols, P denotes the prover and V denotes the verifier.

Proof for Power of $\mathcal{N}$

This protocol proves, given:

Input: $\mathcal{N}$ and u

Private input for P: v such that $u = v^{\mathcal{N}} \mod \mathcal{N}^2$

that u is a power of $\mathcal{N}$ modulo $\mathcal{N}^2$. The sequence of steps is:

1. P chooses a random r mod $\mathcal{N}^2$ and sends to V $a = r^N$ mod $\mathcal{N}^2$.
2. V chooses a random k bit number e and sends e to P.
3. P sends $z = rv^e \mod \mathcal{N}^2$ to V and V checks that $z^{\mathcal{N}} = au^e$ mod $\mathcal{N}^2$ and accepts only if and only if this is the case.

Proof for 1-out-of-2 power of $\mathcal{N}$

This protocol proves, given:

Input: $\mathcal{N}$, u_1, u_2
Private input for P: v_1 such that $u_1 = v_1^N$ mod $\mathcal{N}^2$
that either u_1 or u_2 is a power of $\mathcal{N}$ modulo $\mathcal{N}^2$. The sequence of steps is:

1. P chooses a random r_1 mod $\mathcal{N}^2$. He invokes M on input $\mathcal{N}$, u_2 to get a conversation a_2, e_2, z_2. He sends $a_1 = r_1^N$ mod $\mathcal{N}^2$, a_2 to V.
2. V chooses s, a random t bit number and sends s to P.
3. P computes $e_1 = s - e_2 \mod 2^t$, $z_1 = r_1 v_1^{e_1}$ mod $\mathcal{N}^2$ and sends e_1, z_1, e_2, z_2 to V.
4. V checks that $s = e_1 + e_2 \mod 2^t$, $z_1^N = a_1 u_1^{e_1}$ mod $\mathcal{N}^2$ and, $z_2^N = a_2 u_2^{e_2}$ mod $\mathcal{N}^2$, and accepts if and only if this is the case.

In the above protocol M denotes the honest-verifier simulator for *proof for power of N* protocol above. Using these proofs as building blocks, we present our construction for selecting in Zero Knowledge a subset of candidates belonging to one and only one party.

5 Voting Protocol

The purpose of the vote is to select t candidates from a single party. Suppose there are K parties participating in the election and let each party have L candidates. The purpose of such an election would be to select a governing party as well as the members of the parliament, or elect a mayor of a municipality and the members of the council. The voter has to prove that the vote is valid, which means that it contains at *most* t positive selections and that all those selections are from the *same* party since one should not be able to vote for candidates belonging to deferent parties.

Since the total number of candidates is $K \times L$ we will hold $K \times L$ parallel "yes/no" votes. In this sense the voter will select "yes" for t candidates and "no" for the rest. If we represent "yes" with "1" and "no" with "0", then the encrypted selection for candidate i will be $E_i = g^1 \times r^{\mathcal{N}} = g \times r^{\mathcal{N}}$ if the selection is "yes" or $E_i = g^0 \times r^{\mathcal{N}} = r^{\mathcal{N}}$ is the selection is "no", for some random $r \in \mathcal{Z}_{\mathcal{N}}^*$.

Along with the encryptions of the "yes/no" selections for the $K \times L$ candidates, the voter needs to send proof of the validity of those encryptions. For this purpose the protocol for *1-out-of-2 power of* $\mathcal{N}$ is utilized. Using this protocol, the user proves that either E_i or E_i/g is a power of $\mathcal{N}$ modulo $\mathcal{N}^2$, essentially proving that the vote is either an encryption of "0" or "1". Most importantly,

however, the voter has to prove that the t selected candidates come from the same party. For this purpose the voter needs to generate K proofs, one for each party, proving that either E_{p_i} or E_{p_i}/g^t is a power of $\mathcal{N}$ modulo $\mathcal{N}^2$ using the protocol for 1-out-of-2 power of $\mathcal{N}$. E_{p_i} is the product of the encrypted selections for party i:

$$E_{P_i} = \prod_{\text{Party } i} E_j \ . \tag{3}$$

In this manner the voter proves that he has either selected 0 or t candidates from this party. This is due to the homomorphic property of the cryptosystem, since E_{p_i} will either be an encryption of 0 or t. Notice that the voter reveals no information about the party she has voted for, since she proves for *each* party that she has either selected 0 or t candidates from it, which is essential for the validity of the submitted ballot.

Finally the voter needs to prove that she has selected exactly t candidates. In order to do that, the voter uses the *protocol of* $\mathcal{N}$ *power* in order to prove that $\Pi_i E_i/g^t$ is a power of $\mathcal{N}$ modulo $\mathcal{N}^2$, where $\Pi_i E_i$ is the product of all the encrypted selections. Due to the homomorphic property of the cryptosystem $\Pi_i E_i$ will be an encryption of "t" and $\Pi_i E_i/g^t$ will be a power of $\mathcal{N}$.

Having proven the above properties for the submitted encrypted selections the user successfully shows that she has selected exactly t candidates from a single party and no candidates from the rest of the parties. This is the case since she proves that for every party she has selected t or 0 candidates and that she has selected t candidates in total.

It is easy to generalize the above protocol to allow up to t selections from a single party by adding t "dummy candidates" to the L candidates of each party. The voter will then place the selections she does not wish to use on the "dummy candidates". In the same manner we can accommodate for blank votes be creating a dummy party with t candidates on which the selection will be placed.

6 Voting System Implementation

We will use a hash function h in order to make the proofs used by the above protocol non-interactive according to the Fiat-Shamir [9] heuristic. We will also assume that an instance of threshold version of Paillier's scheme with public key $(\mathcal{N}, g)$ has been set up, with A_i's being the decryption servers. We will also assume that $\mathcal{N}^2 > M$, where M is the number of voters, since $\mathcal{N}$ can always be chosen large enough to satisfy this inequality.

As we have mentioned in the introduction, we will use a general model for election described in [4], which we describe briefly: We have a set of voters $V_1, V_2 \ldots, V_M$, a bulletin board B and a set of tallying authorities $A_1, A_2 \ldots, A_M$. The bulletin board operates as follows: Every voter can write to B and no message can be deleted from B once it has been written there. Everyone can access the messages in B and can identify the origin of each message. This can

be implemented in a secure way using existing public key infrastructure. Also assume that we have K parties $P_1, P_2 \ldots, P_K$.

6.1 Voting Procedure

The voting procedure can be summarized with the following steps:

1. Each voter V_i decides on his votes 0 for "no" 1 for "yes" for the $K \times L$ candidates, calculates $E_{ij} = E(v_{ij})$ and creates proofs:
   - Proof that E_{ij} or E_{ij}/g is an $\mathcal{N}$ power modulo $\mathcal{N}^2$ for all $j \in \{1 \ldots K \times L\}$ and
   - Proof that $\prod_{j \in P_k} E_{ij}$ or $\prod_{j \in P_k} E_{ij}/g^t$ is an $\mathcal{N}$ power modulo $\mathcal{N}^2$) for all $k \in \{1 \ldots K\}$
   - Proof that $(\prod_{j=1}^{K \times L} E_{ij}/g^t$ is an $\mathcal{N}$ power modulo $\mathcal{N}^2)$

   She writes the encrypted votes E_{ij} and all the proofs in B.
2. Each authority A_k sets $\prod E_j = 1$ for all $j \in \{1 \ldots K \times L\}$. Then for all voters i
   - Checks the proofs in B for voter V_i and if they are valid sets $\prod E_j = (\prod E_j) \times E_{ij} \bmod \mathcal{N}^2$.
   - Finally A_k executes its part of the threshold decryption protocol, using $\prod E_j$'s as input ciphertext and writes its result to the bulletin board B.
3. From the messages written by the tabulation authorities in B one can now reconstruct the final tally $\prod E_j$ for $j \in \{1 \ldots K \times L\}$. Clearly $\prod E_j = \prod_i E(v_{ij}) = E(\sum_i v_{ij})$. Therefore the decryption results will be $\sum_i v_{ij} \bmod \mathcal{N}^2$ for candidates $j \in \{1 \ldots K \times L\}$ which is $\sum_i v_i j$ since $\mathcal{N}^2 > M$. One can also easily calculate the votes for each party by summing the votes that the candidates of each party have received and divide them by t:

$$Votes_{Party_k} = \sum_{j \in Party_k} v_{ij}/t \ . \tag{4}$$

The security of this protocol follows from the security of the sub-protocols used and the semantic security of Paillier's Cryptosystem.

6.2 Complexity Analysis

In the analysis that follows we will denote with C the total number of candidates that participate in the election. Without loss of generality we examine the case of K parties with L candidates each and an election that allows up to t selections within a party, so C will be equal to $K \times (L + t)$.

The voter generates C proofs of $1 - out - of - 2$ *power of* $\mathcal{N}$, one for each candidate and K proofs of $1 - out - of - 2$ *power of* $\mathcal{N}$, one for each party. Thus the voter needs to generate $C + K$ proofs in total which is clearly $O(C)$. If k is the bitlength of $\mathcal{N}$ then evidently the size of a vote in this voting protocol is $O(C \times k)$. The same holds for the time needed for the computation and verification of the proofs.

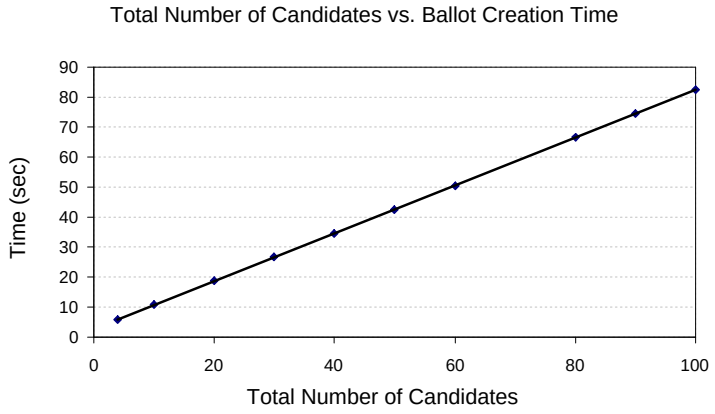

Fig. 1. Proof Creation Time versus Total Number of Candidates participating in the election

We have implemented a system using the above protocol and made measurements of the running time on a Pentium 4, 2.53GHz machine. In Figure 1 the time needed to create the Zero Knowledge proofs is displayed versus the total number of candidates in the elections. The system behaves linearly to the number of candidates and the computation time is acceptable even when a much larger number of candidates (> 100) is used. For the implementation Java was used, which means that native code will reduce the computation time even further. For the tally computation $O(C \times M)$ verifications of proofs need to be performed, where M is the number of voters as well as $O(C)$ decryption operations on the products of the individual votes. The time to verify the proofs for a single voter is less than that of the proofs' creation time. These operations, as well as the decryption operation can be done offline after the end of the elections without increasing the voter's waiting time.

The size of the ballots created, as we mentioned above, increases linearly to the number of candidates in the elections. In Figure 2 we show how the ballot size varies with respect to the total number of candidates in our system. As we expected the ballot size doubles when the number of candidates doubles.

We should note that our proposal is not limited to parties that have the *same* number of candidates each. We have used this model so far to simplify the description of the system and not to complicate the notations used in the Zero Knowledge proofs. Our proposal can be applied to elections with *parties that have an arbitrary number of candidates* each and the analysis of the system complexity we made above still holds.

Finally, we provide a formula for the ballot size in our system. In what follows, C is the number of candidates in the system which are distributed (in any manner) in K parties. Let $|H|$ denote the size of the hashed commitments

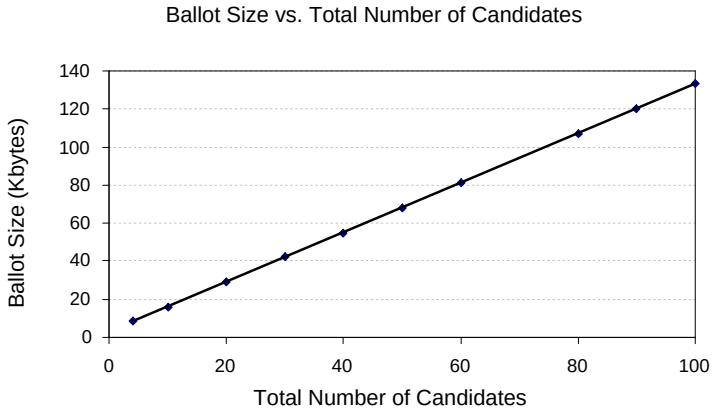

Fig. 2. Ballot Size (in Kbytes) versus Total Number of Candidates participating in the election

and $|\mathcal{N}^2|$ the size of the modulus used in the Paillier cryptosystem. The size of a vote is then

$$(5|\mathcal{N}^2| + 2|H|) \times C + (4|\mathcal{N}^2| + 2|H|) \times (K + 1) \ . \tag{5}$$

Evidently the ballot length is $\mathrm{O}(C + K)$ and since in most cases $C >> K$ we can say that our system behaves linearly to the number of candidates. In practical applications we may choose $|H| = 80$ and $|\mathcal{N}^2| = 2048$. For an election with 20 candidates and 3 parties the size of a vote is about 30Kbytes.

7 Conclusions and Future Research

In this paper we have presented a methodology for proving in Zero Knowledge the validity of a selection of t elements from one out of k sets of n elements each. We have used this methodology in order to construct an election scheme that can provide more complex ballot options than current existing ones. In particular, our implementation can host elections involving a number of parties each consisting of an *arbitrary number* of candidates with the voter having to select a subset of the candidates from a *single party*, without degrading security or efficiency. Furthermore, the computational and communication complexity is linear with respect to the number of candidates and the proposed voting system satisfies all necessary security requirements.

In our current research, we are examining the possibility to make the complexity of our system (ballot size and computations) proportional to the number of parties *plus* the number of selections the voter is requested to make. This may be achieved possibly by representing a selection as in [8] and coming up with a

new set of Zero Knowledge proofs which can correlate the selection with a single party.

References

1. J. Benaloh. Verifiable Secret Ballot Elections. PhD thesis, Yale University 1997
2. J. Benaloh and D. Tuinstra. Receipt-free secret ballot elections. In Proc. 26th Symposium on the Theory of Computing (STOC), pages 544-553 ACM 1994
3. R. Cramer, Y. Frankel, B. Schoenmakers and M. Yung. Multi-Authority Secret-Ballot Elections with Linear Work. In Eurocrypt '96, LNCS 1070, pages 72-83. Springer-Verlag, 1996
4. R. Crammer, R. Gennaro and B. Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. In Eurocrypt '97, LNCS 1233, pages 113-118. Springer-Verlag, 1997
5. P. Paillier. Public-Key Cryptosystems Based on Discrete Logarithm Residues. In Eurocrypt '99, LNCS 1592, Springer-Verlag, 1999
6. I. Damgraad and M. Jurik. A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In PKC '01, LNCS 1992, pages 119-136, Springer-Verlag, 2001
7. P. A. Fouque, G. Poupard and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. In Financial Crypto '00, LNCS, Springer-Verlag, 2000
8. O. Baudron, P.-A. Fouque, D. Pointcheval, G. Pouparde, and J. Stern. Practical Multi-Candidate Election System. Proceedings of the ACM Conference on Principles on Distributed Computing, August 2001, Philadelphia, USA
9. A. Fiat and A. Shamir. How to Prove Yourself: practical solutions of identification and signature problems. In Crypto '86, LNCS 263, pages 186-194, Springer-Verlag, 1987
10. A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections, Advances in Cryptology - AUSCRYPT '92
11. 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, Springer-Verlag, 1992
12. I. Damgard and M. Koprowski. Practical Threshold RSA Signatures Without a Trusted Dealer. In Eurocrypt '01, LNCS, Springer - Verlag, 2001
13. P. Fouque and J. Stern. Fully Distributed RSA Signatures under Standard Assumptions. 2001