How to Prevent Buying of Votes in Computer Elections

Valtteri Niemi[1] and Ari Renvall[2]*

[1] Department of Mathematics and Statistics, University of Vaasa, 65101 Vaasa, Finland
valtteri.niemi@uwasa.fi
[2] Department of Mathematics, University of Turku, 20500 Turku, Finland
ari.renvall@utu.fi

Abstract. We present a new voting scheme. Our scheme is the first one which satisfies both of the following properties: each voter can check that her voting strategy is counted correctly but she cannot prove the voting strategy to anybody except the full coalition of all interest groups involved in the voting. The latter property means that buying of votes is not possible, since the potential buyer does not know whether she gets what she is paying for!

1 Introduction

Computerized elections and votings in general constitute a main application area of cryptographic protocols. Many elegant voting protocols exist [1], [2], [6], [8], [9] and they satisfy the most important requirements including secrecy of votes and correctness of results. Of course, there are many differences in type and level of security, efficiency etc.

However, all these protocols are unsatisfactory in one sense: they allow buying of votes. In traditional voting systems this is prevented in a simple manner, i.e. the voter cannot prove to a potential buyer that she/he indeed cast the vote as agreed. In all systems mentioned above any voter can invite anybody to witness casting of vote by remote terminal. The voter can even let the buyer cast the vote instead of her/him.

It can be argued how severe this problem is. Someone might have the opinion that buying of votes would be very rare and has no real impact on the result. We just want to point out that if it is believed that nobody bothers to buy votes in large scale then it might be reasonable to believe also that nobody bothers to corrupt a neutral trusted center with big money. In the latter case there is hardly any need for complicated protocols at all, a simple encryption of votes and decryption by a reliable center would be secure enough.

In [7] the problem was attacked with partial success. A detailed protocol was constructed with one essential shortcoming: it was assumed that one fixed party involved is not interested in buying votes. In the end of [7] an idea was given for

* Research supported by The Academy of Finland, grant 11281

a protocol without this shortcoming. The present paper develops the idea and we construct a protocol which prevents the voter from proving her/his voting strategy to anybody.

The basic idea is to add a preliminary registration of voters. It takes place in physically controlled circumstances and during it each voter gets some information that is needed in the actual voting. This information can, of course, be given to anybody but there is no way of checking its correctness. Indeed, the correctness is proved to the voter in zero-knowledge in a secure and private registration booth.

At first glance, it seems that a registration that requires everybody to go to some place destroys the whole idea of computerized voting. However, this is needed very seldom and only once for many different votings. Also, it is clear that some kind of physical identification of voters is needed anyway.

The paper is structured as follows. Next section gives some basic concepts and an informal description of the protocol. Section 3 contains the protocol while it is discussed in section 4.

2 An Informal Description of the Protocol

In this section we discuss the main ideas of our voting scheme. Remember that the goal is to prevent the possibility to prove one's voting strategy. Denote the voters by V_i, their votes by v_i and the corresponding voting strategy by $[v_i]$.

In traditional *balloting booth elections* connection between V_i and v_i is physically broken by scrambling v_i with a number of other votes. As a consequence V_i cannot prove which is her vote. Therefore in this setting we may choose simply $v_i = [v_i]$. In computerized secret ballot elections this is obviously not possible. In common solutions (like [1], [2], [6], [8], [9]) the voting strategy is encrypted in some way or the vote is cast anonymously together with an *eligibility token*. However, in all schemes presented so far the voter has always been able to open the encryption or to prove that a given eligibility token is the one she used. This means that the voter has been able to prove her voting strategy.

In our scheme the vote is also essentially of the form $v_i = ([v_i], e_i)$, where e_i is an eligibility token. But we construct the token in such a way that V_i can not prove the validity of e_i. In order to do this we use a special type of permutation.

Definition *A permutation f is a* zero-way permutation, *if there exists a trapdoor s and an injective function g such that*

1. *both f and f^{-1} can be efficiently computed if one knows s.*
2. *neither f nor f^{-1} can be efficiently computed if one knows only $g(s)$.*

It follows from the definition, that s cannot be computed from $g(s)$. Thus, one can commit to a zero-way permutation by disclosing $g(s)$.

It seems plausible to conjecture that zero-way permutations exist if trapdoor permutations exist. For instance, if $f = f_1 \circ f_2^{-1}$ where f_1 and f_2 are trapdoor

permutations over a set X, then computing both f and f^{-1} requires to invert either f_1 or f_2. There are also other methods to obtain (candidate) zero-way permutations.

The eligibility token of our scheme is now just an element of a set $f^{-1}(Y)$ where $f : X \to X$ is a zero-way permutation chosen by the government of the voting and $Y \subseteq X$ is a suitable set. Of course, it must be computationally infeasible to generate valid tokens without the help of C. For this reason, Y should be small enough. On the other hand, Y should not be too small in order to guarantee that no two voters could get identical tokens. For example, Y could consist of all such binary strings that contain a long sequence of zeros in the middle while X contains all strings (of fixed length).

The idea of our scheme is that in a preliminary registration phase C gives each legitimate voter one valid token. The voter gets only a zero-knowledge proof of the validity of her token and can thus not transfer her conviction of the validity.

In the voting phase the voters are now allowed to cast as many votes as they wish. However, they can construct only one vote with a valid token (multiple use of same token is not allowed).

In the final phase of the scheme C computes the tally of the election. C collects the votes corresponding to the same strategy and then computes the number of valid tokens associated to them. This is done so that the the validity of any single token is never checked. Thus, the number of possible false votes with invalid tokens will be counted but no-one will find out which votes they are.

If we perform the protocol the way we described it, one severe problem arises. Even if we can not prove our voting strategy to outsiders, we can prove it to C (in fact C could find it out by itself). We solve this by dividing C to subgovernments $C_1,\ldots,C_n$. For example, all the candidates (or their representatives) might act as subgovernments, as in [6]. The basic idea is that every potential interest group can trust at least one of the subgovernments. (The reliable one may be different for each interest group.)

The zero-way fuction f must then be constructed in such a way that each C_j has a secret share s_j needed to compute f and f^{-1}. The problem of generating a valid eligibility token in the registration phase is then solved by applying a protocol for multiparty computation of [4]. The participants of the protocol are the voter V_i and every C_j. As a result V_i obtains a valid token as a private output. If we modify the protocol slightly V_i cannot prove to any proper subcollection of $\{C_1,\ldots,C_n\}$ what the output was.

In the final phase where the tally of the election is computed we need to apply similar multiparty computation protocol.

3 The Protocol

In this section we give a more detailed description of our protocol. The protocol consists of four phases. For simplicity, we call subgovernments C_j candidates in the sequel.

3.1 Preliminary Phase

This phase occurs (e.g.) once in a year. During it candidates make necessary preparations for each (possible) voting of the year. The idea is to build all the computational circuits needed later in the year.

1. Candidates agree and fix some family of zero-way and trapdoor permutations. For instance, a modulus p and a generator g of $\mathbf{Z}_p^*$ is chosen for permutations based on discrete logarithm. (This example is also used later in the protocol but other permutations like RSA-based ones could be chosen as well.)
2. For each future voting, every candidate C_j $(j = 1, ..., n)$ generates a random number s_j. They also commit to these numbers by publishing $g^{s_j} \pmod p$. The product $s = s_1 \cdots s_n$ serves as the secret key of the zero-way permutation $f : x \mapsto x^s \pmod p$, see [3], [7]. No proper subset of candidates can compute either f or f^{-1}.
3. Candidates perform a multiparty computation to obtain the element $g^s \pmod p$. This is used as their public key in the El Gamal cryptosystem. (For details of the system, consult [5] or some standard textbook, e.g. [10].) Also here s serves as the trapdoor information. The resulting encryption function is denoted by E and decryption function by D.

3.2 Registration Phase

In this phase each voter obtains an eligibility token for each future voting. (Of course, the set of votings for which token is given may vary with respect to, e.g. age and address of the voter.) This phase occurs also once in a year.

1. The voter is physically identified.
2. She/he goes into a physically secured booth that has private communication lines to each candidate.
3. For each voting, candidates and the voter execute a multiparty computation which gives a private output to the voter. The output is a number $f(x) = x^s \pmod p$ where x is of a very special form. More specifically, assume p contains $4n$ bits. Then x is OK if $x = z_1 \ldots z_{4n}$ where $z_i = 0$ for $i = n+1, n+2, \ldots, 3n\}$. (Other forms may do as well.) The number $f(x)$ is called an eligibility token. It is important that the voter cannot prove validity of the token to anybody. That means that the multiparty protocol of [4] must be modified in the following way:
   (a) Each candidate chooses random numbers x_j and b_j. Her private input for the protocol consists of x_j, b_j and s_j.
   (b) Each candidate sends (privately) b_j to the voter. At the same time she gives a private zero-knowledge proof that convinces the voter of the correctness of b_j. This can be easily added to the protocol of [4] since the candidate

must commit to b_j anyway using some bit commitment scheme. (We do not go into the details of this modification, since description of the protocol in [4] would be needed and it is quite lengthy.)
(c) Output of the computation is $f(x) \oplus b_1 \oplus \ldots \oplus b_n$ where x is obtained by xoring all the x_j's and adding the 0-sequence in the middle. The voter obtains the token by xoring the output by $b_1 \oplus \ldots \oplus b_n$. She has no way of proving the value of $f(x)$ to anybody (except to the full coalition of all candidates). On the other hand, the probability that a random number is a valid token is 2^{-2n}.

3.3 Voting Phase

During each voting everybody has a chance to send (possibly anonymously) messages to a public file F. Each voter sends $E([v_i], f(x))$ where a legitimate vote is encrypted by the public key of that particular voting. If the message does not show up in F the voter simply sends it again. It is worth noting that nobody is prevented from sending false messages anyhow.

3.4 Counting Phase

After every interested and legitimate voter is happy with the public file F candidates perform a multiparty computation to calculate the tally of the voting. The computation has a public input F and private inputs s_j.

1. Entries in F are decrypted and the resulting list of pairs of the form (voting strategy, token) is used as a secret input in the next step.
2. For each possible voting strategy the number of valid tokens associated to it is counted. This can be done, since all candidates together can compute f^{-1}-image of the token and check that it has the 0-sequence in the middle. It is important that the result of the validity check is solely used as an intermediate output and a secret input for the final counting circuit which gives a non-negative integer as an output.
3. Tally of the voting is the public output of the computation and anybody can check the correctness of the computation as explained in [4].

Nobody (except a full conspiracy of all candidates) obtains any information of individual entries in the public file F other than what can be derived from the tally alone.

Note that a voter may join a valid token to several voting strategies! There are several ways of preventing this. For example, all different strategies with the same token could be rejected (as a penalty for attempted misuse of the protocol). Another possibility is to check (of course, inside the multiparty computation) which of the pairs with the same token came first (or last if we want to allow a last-minute change of opinion).

4 Some Remarks

Our protocol uses heavily multiparty computations. However, operations needed are quite simple like modular exponentiations or counting of valid tokens. That means the number of gates in the circuits is not extremely large. On the other hand, the (potentially enormous) number of false votes raises the complexity of the protocol considerably although the growth is linear. Altogether, our protocol is clearly impractical. Special-purpose multiparty protocols could make a change in this matter. The main goal for future research is to construct an efficient protocol that provides the same features as our protocol.

Many minor modifications can be made in the protocol. In small scale votings it is quite possible that voters themselves act as subgovernments. Then the first two phases can be combined.

As already mentioned, there is much freedom in choosing the proper zero-way and trapdoor permutations. The main criterion is their suitability for distributed computations with private inputs. Also, it is by no means necessary that the chosen zero-way and trapdoor permutations are linked anyhow.

Encryption of (strategy,token)-pairs is needed since otherwise an active intruder could simply change the strategy and leave the token as it is. However, it could be skipped if each voter gets several tokens for each voting and these tokens are used for coding the voting strategy. More specifically, a fixed number of 0-tokens (resulting from binary strings with long 0-sequences as above) and 1-tokens (resulting from strings with 1-sequences) is given to each voter. Nobody except the whole collection of candidates can make any difference either between tokens and non-tokens or between 0-tokens and 1-tokens. The voting strategy is coded by the order of the tokens in such a way that even if the strategy of some voter is guessed the order of the tokens cannot be changed to produce a code for some other strategy. (For example, first tokens code the strategy in some straight-forward numbering system and most of the tokens are put in one of many possible orders that are associated to this particular strategy beforehand.) Thus, an active intruder could destroy the message at best and in case it happens, the voter can try again, as explained above.

References

1. J. Benaloh: Verifiable secret-ballot elections, Ph.D. thesis, Yale university, Technical report 561, (1987).
2. D. Chaum: Untraceable electronic mail, return address, and digital pseudonyms, *Comm. of ACM*, **24** (1981), pp. 84–88.
3. D. Chaum: Zero-knowledge undeniable signatures, *Proc. EUROCRYPT '90, Springer LNCS 473*, (1991), pp. 458–464.
4. D. Chaum, I. Damgård, J. van de Graaf: Multiparty computations ensuring privacy of each party's input and correctness of the result, *Proc. CRYPTO '87, Springer LNCS 293* (1988), pp. 87–119.
5. T. El Gamal: A public key cryptosystem and signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory* **IT-31** (1985), pp. 469–473.

6. K. Iversen: A cryptographic scheme for computerized general elections, *Proc. CRYPTO '91, Springer LNCS 576* (1992), pp. 405–419.
7. V. Niemi, A. Renvall: Cryptographic protocols and voting, *Proc. Results and Trends in Theoretical Computer Science, Springer LNCS 812* (1994), pp. 307-316.
8. H. Nurmi, A. Salomaa, L. Santean: Secret ballot elections in computer networks, *Computers and Security* **10** (1991), pp. 553–560.
9. C. Park, K. Itoh, K. Kurosawa: Efficient anonymous channel and all/nothing election scheme, *Proc. EUROCRYPT '93, Springer LNCS 765* (1994), pp. 248–259.
10. A. Salomaa: Public-Key Cryptography, Springer-Verlag Berlin Heidelberg 1990.