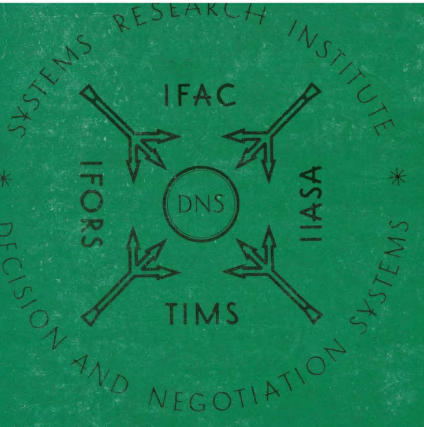

IFAC/IFORS/IIASA/TIMS

The International Federation of Automatic Control
The International Federation of Operational Research Societies
The International Institute for Applied Systems Analysis
The Institute of Management Sciences

SUPPORT SYSTEMS FOR DECISION AND NEGOTIATION PROCESSES

Preprints of the IFAC/IFORS/IIASA/TIMS Workshop
Warsaw, Poland
June 24-26, 1992

Editors:

Roman Kulikowski
Zbigniew Nahorski
Jan W. Owsiński
Andrzej Straszak

Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland

VOLUME 2:

Names of first authors: **L-Z**

SYSTEMS RESEARCH INSTITUTE, POLISH ACADEMY OF SCIENCES

SUPPORT SYSTEMS FOR DECISION AND NEGOTIATION PROCESSES
Preprints, IFAC/IFORS/IIASA/TIMS Workshop, June 24-26, 1992, Warsaw, Poland

Conducting Secret Ballot Elections in Computer Networks: Problems and Solutions

Hannu Nurmi
Department of Political Science
University of Turku
20500 TURKU
Finland

and

Arto Salomaa
Academy of Finland and Department of Mathematics
University of Turku
20500 TURKU
Finland

The research reported in this paper has been supported by the Academy of Finland grants 1071178 (Nurmi) and 1071041 (Salomaa).

Abstract: We discuss problems related to devising a secret balloting system with the following properties: 1. all eligible voters and they only may vote, 2. all ballots are secret, i.e. do not reveal the identity of the voter, 3. all voters may check whether their ballots have been correctly assigned, 4. the voters may revise their ballots within a predetermined time, and 5. errors in ballot assignment can be corrected within a predetermined time.

Keywords: cryptographic protocols, secret ballot, public-key cryptosystems, voting

1. Introduction

Periodic elections are an institution that can be viewed as a necessary - albeit insufficient - condition of democracy. During their millenia long history, elections have assumed many forms, but at least *prima facie* the goal has always been the same: to reveal the will of the electorate in matters to be decided. To what extent that goal can be reached depends on the particular voting procedure being used and the opinions of the individual voters. In particular, it depends on the willingness of the latter to reveal their true opinions on the issue at hand.

Secret ballot is one crucial step in the way of securing that the voters express their true opinions in voting. In other words, ballot secrecy is a necessary condition for a true

revelation of preferences. Without ballot secrecy the voters may not feel free to express their views if they fear that they will later be ridiculed, despised, or persecuted for them. Thus far the practical method for guaranteeing ballot secrecy has been to cast the ballots in designated places under the surveillance of election officers who protect the privacy of the voters.

In this paper we shall focus on the possibility of conducting secret ballot elections in computer networks. We shall outline a set of conditions that we want our balloting system to satisfy and discuss the tradeoffs between the outlined system and the existing secret balloting schemes. Throughout this paper we shall use the expressions "to support a candidate" and "to adopt a voting strategy" interchangeably. For the applicability of the protocols introduced, the distinction is of no consequence.

For other - essentially different and more complicated - secret balloting systems devised for use in computer networks as well, the reader is referred to Benaloh (1987), Chaum (1988) and Iversen (1991). Our own earlier work is reported in Nurmi & Salomaa (1991a), Nurmi & Salomaa (1991b) and Nurmi, Salomaa & Santean (1991).

2. Preliminaries

The secret balloting systems discussed in this paper are based on public-key cryptosystems (see Diffie & Hellman 1976; Salomaa 1990). In these systems, each participant i has two keys: a public encryption key, denoted by e_i, and a private decryption key, denoted by d_i. Consider a plaintext message m. Its numerical encoding is denoted by w. Messages w from participant i to participant j are assumed to satisfy the following:

$d_j(e_j(w)) = e_j(d_j(w)) = w$.

All current public-key cryptosystems are based on the assumption that, given e_i and $e_i(w)$, it is computationally intractable for anyone else but a person who knows d_i to recover w. On the other hand, should one know d_i, the computation of w from $e_i(w)$ is easy, e.g. can be performed in time that is bounded from above by a polynomial function of the "size" of $e_i(w)$.

3. First Problem: Fighting the Eavesdroppers

A quite basic requirement for any secret balloting system is that voter j cannot find out which alternative or candidate voter i votes for. We thus insist that j faces a computationally

intractable task in trying to recover i's vote from the message i sends to the ballot counting system B. *Prima facie*, it would seem sufficient that i sends his vote w in encrypted form using B's public key: $e_B(w)$. This will, however, not do, as i's vote can be determined by j through encrypting - by using B's public key - all possible votes and comparing the results with the message that i sends to B. However, in cases where the source of the message - in our example i - is not recoverable from the message itself, the simple encryption using B's key would be adequate.

In more general settings we require that w is not recoverable from i's message to B by trial and error. To accomplish this, i's vote is sent in a hashed form: $e_B(ptw)$ where p is a large number chosen by i and t is a number given to i by the eligibility checking system C. Here the result of hashing is denoted simply by juxtaposition ptw.

4. Second Problem: Checking the Eligibility

Obviously, no voter should be able to cast more than one valid vote in any given election. Moreover, only eligible voters should be able to vote. To satisfy these requirements, an eligibility checking system C is set up. The following protocol is then performed:

1. Voter i sends C the numerical encoding m of "I'm i and would like to vote" in the form of the following pair:
$e_C(d_i(i))$, $e_C(d_i(m))$.
Here i denotes i's name in numerical form.
2. Upon receiving the message, C first decrypts it using its private key to obtain $d_i(i)$ and $d_i(m)$. C then uses i's public encryption key e_i to recover i and m. The former part of the pair is convenient to speed up the recovery of the right encryption key. If the number of voters is relatively small, the first part of the pair can be deleted since C can find out i's identity by trying each public key in turn to determine which key produces a sensible plaintext message.

This simple protocol guarantees that C can be sure that the voter approaching it is i. Upon receiving the message, C determines if i is an eligible voter. If he is, his name is removed from the list of eligible voters and the following message is sent to i: $e_i(t)$ where t is a large prime number chosen by C. Otherwise, a rejection is issued.

5. Third Problem: Preventing the Collusion of the Authorities

If one could count on the benevolence of the election authorities, one would not need both systems B and C. One would do with B only. (In fact, we have discussed elsewhere a secret balloting system in which one system performs the roles of B and C (see, Nurmi, Salomaa & Santean 1991)). However, everyone knows that fraudulent elections have been conducted. Thus, we cannot rely on on the authorities' willingness to keep the ballots secret. What if the systems collude? More specifically, wouldn't it be possible for C to inform B that voter i just registered and was given the message t? It would, indeed. The crux of the above arrangement is that the message t is given to every eligible voter. Thus, even with C's assistance, B is unable to recover i's vote from the message that i sends to B.

We are, however, not quite happy with the above arrangement as it allows in principle any legitimate voter to sabotage the election by casting several votes. Of course, the fact that someone is trying to cheat will eventually be spotted by tallying the votes, but the identity of the voter cannot be determined. A more complicated protocol could, however, be envisaged for the interaction between i and C. We now briefly outline that protocol.

Before the election, systems B and C have agreed upon a large set L of large numbers. The cardinality of L should be much larger than that of the set of voters. Once voters 1, ..., n have indicated their willingness to vote and C has verified their eligibility and determined that they have not yet voted, C offers the voters the choice of one of the numbers in L so that after i has chosen a number, say l_i, from L, C does not know which particular element from L was chosen by i. Moreover, each i chooses precisely one number from L (see Salomaa and Santean 1990 and Nurmi, Salomaa and Santean 1991 for details).

Now, instead of the number t mentioned above, each voter i uses his own l_i to perform a cryptographic hash of his vote. Thus, his message to B takes the following form:

$$e_B(l_i), e_B(l_i w). \qquad (5.1.)$$

From this message B can easily determine that i is a legitimate

voter (since l_i is in L) and that his vote is w.

It may, however, turn out that two or more voters have chosen the same number l_j. We shall return to this possibility later on.

6. Fourth Problem: How to Check Your Ballot?

Once the ballots have been cast, system B publishes the election results by indicating for each candidate - or more generally for each voting strategy - the identification numbers l_i of those voters who supported that candidate or chose that voting strategy. Thus, voter i, for example, verifies that B has assigned his vote to the candidate he voted for, candidate x, say. To do this, voter i looks at the list of the identification numbers of x's supporters and finds out whether l_i is there. (The identification numbers may be published in an increasing order of magnitude to make the voters' verification task easier.)

The numbers chosen by voters from L can at this stage be published in plain form, as nobody else but i is able to connect i with l_i. The possibility to verify that one's vote has been correctly assigned is definitely an improvement over current electoral methods.

7. Fifth Problem: What If Something Goes Wrong?

While the preceding stages already exclude the possibility of some forms of electoral fraud remaining undetected, they do not enable the voters to protest without jeopardizing the ballot secrecy. To solve this problem, a slight complication in the above voting protocol is called for.

Upon receiving the identification number l_i from C, voter i sends B the the following pair:

$$e_B(l_i),\ e_B(f_i(l_i,\ w)) \qquad (7.1.)$$

where f_i is a cryptographic hash function chosen by i. It is a one-way function such that given $f_i(x,y)$ it is computationally intractable to find x and y. Moreover, given $f_i(x,y)$ and x, it is computationally intractable to find y, whereas given $f_i(x,y)$, x and f_i^{-1}, y can always be easily computed. We require that the first component be sent in an encrypted form as otherwise it could be utilized by eavesdroppers who could capture l_i from i's message to B and pretend to be legitimate voters who "accidentally" have the same identification numbers as i.

Once B has received (7.1.) from i, it issues a "receipt":

$f_i(l_i,w)$. (7.2.)

At this stage B does not know i's vote (nor, of course, i's identity). Voter i now sends B the following message:

$e_B(l_i, f_i^{-1})$. (7.3.)

Hence, upon receiving (7.3.), B knows the vote of the voter whose identification number is l_i. Of course, B does not know who this voter is.

Now, in this more complicated protocol, B publishes the election results by indicating for each voting strategy v the list of those $f_i(l_i, w)$'s for which $w = v$. The role of (7.2.) is to enable the voters to check whether their message has reached B. If no receipt is issued by B, i can demonstrate B's error (without revealing his vote) by publicly showing that the latter part of (7.1.) in decrypted form - which amounts to (7.2.) - is not among the receipts issued by B.

Supposing that B has - intentionally or otherwise - misplaced i's vote, i can issue a protest to B in the following form:

$e_B(l_i, f_i(l_i,w), f_i^{-1})$. (7.4.)

The first component of (7.4.) demonstrates that i is a legitimate voter, the second that he knows the hash function, and the third that he knows the inverse of the hash function. Only i can know all these.

Let us now return to the problem of two or more voters getting the same identification number l_i. The fact that several voters may share the same identification number makes it possible in principle that all legitimate voters may cast several ballots just by resorting to different hash functions each time they approach B with (7.1.). The tallying of votes, of course, makes it impossible for the cheating to go unnoticed, but the culprit cannot be identified.

It should be emphasized, however, that the voters do not have incentives to leak their own identification numbers to others. The fact that two voters have the same l_i does not mean that the receipt (7.2.) issued by B would be different. Hence, without voter collusion only B would know that several voters have the same l_i. Should voter i find out that he and voter j have the same l_i, the only way he could benefit from this information is by casting two ballots and thereby sabotaging the elections (as "too

many" ballots would be cast). Voter j would definitely not benefit from informing i that they both have the same identification number. Thus, the incentives for voter collusion are absent.

8. Sixth Problem: Having Second Thoughts

Suppose now that voter i is for some reason unhappy with his voting strategy or choice of candidate. The following arrangement enables him to change his mind within a predetermined time.

1. Voter i sends B the message:

$$e_B(l_i), e_B(h_i(l_i, w')) \qquad (8.1.)$$

where h_i is another cryptographic hash function chosen by i and w' is his revised voting strategy or candidate choice.

2. B issues the receipt:

$$h_i(l_i, w'). \qquad (8.2.)$$

3. Voter i sends B the following:

$$e_B(l_i, h_i^{-1}). \qquad (8.3.)$$

4. B computes - on the basis of the information given by i in stages 1. and 3. - w', removes $f_i(l_i, w)$ from the list of the voters choosing w, and adds $h_i(l_i, w')$ to the list of voters choosing w'.

After stages 1. - 4. the voters may again check whether their revised votes have ended up in favour of the candidate they - after reconsidering the issue - want to support.

9. Possibilities for Cheating or Tampering

All messages to systems B and C are sent in encrypted form. Finding out the content of those messages is thus tantamount to breaking RSA (see Rivest, Shamir & Adleman 1978; Salomaa 1990). With a relatively small number of possible messages, however, the eavesdropper may find out the content of a given message by simply encrypting all possible messages and comparing them with the observed one. When i contacts C to receive an identification number, the secrecy of messages is guaranteed by the use of the secret-selling-of-secrets protocol.

In the following we shall once more go through all the messages sent through the network to evaluate the difficulty of either eavesdropping or tampering with legitimate ballots.

Message (5.1.) from i to B: the use of B's public encryption key makes the task of recovering w from the message computationally difficult. More specifically, if RSA is used, the task amounts to breaking RSA. Could an illegitimate person then capture the

message from i to B, substitute his own vote for i's vote and send the end result to B without the latter's noticing that something fishy is going on? No, since recovering l_i from $e_B(l_i)$ is computationally intractable.

Message (7.1.) from i to B: message w cannot be be recovered from (7.1.) even by an eavesdropper who knows who sent (7.1.). The reason is that the message is hashed by a function f_i. In fact, not even B can decrypt w from (7.1.). Tampering with i's ballot by capturing $e_B(l_i)$ and substituting one's own hash function and vote for i's f_i and w would not succeed, either, since l_i is needed to recover w.

Message (7.2.) issued by B: this contains the value of the hash function f_i for argument value pair (l_i,w). By assumption that f_i be one-way, neither l_i nor w is recoverable from (7.2.).

Message (7.3.) from i to B: this contains the inverse of the hash function thus far known to i only. The encryption with e_B guarantees that the inverse function cannot be decrypted by eavesdroppers.

Message (7.4.) from i to B: in this message i protests that his vote has not been properly allocated. Voter i's vote cannot be recovered from the message unless the eavesdropper knows both d_B, l_i and f_i^{-1}. An illegitimate voter might wish to be able to use i's ballot by intercepting, i.e. by capturing $e_B(l_i)$ from (7.1.) and then using this information to substitute his own hashed vote for i's. The encrypted version of l_i will not be enough, however. Of course, (7.2.) gives the interceptor also the middle element of (7.4.) as well. The tricky part is the inverse of the hash function. This is only known to i.

Message (8.1.) from i to B: in this message i expresses his wish to change his mind about how to vote. Its first component reveals that he is a legitimate voter. It is expressed in encrypted form and is thus unaccessible to eavesdroppers. However, an interceptor might capture the first component and try to substitute his own "new" hash function, its inverse and "new" vote for those of the legitimate voter. This, however, will only be possible if the interceptor can decrypt $e_B(l_i)$ which is computationally intractable.

Message (8.2.) issued by B: this has the same properties as the message (7.2.) discussed above.

Message (8.3.) from i to B: this, in turn, has the same properties as (7.3.) discussed above.

10. Concluding Remarks

In the preceding we have outlined a protocol with a few variations for conducting secret balloting in computer networks using public-key cryptography. The outlined system is based on the idea that some forms of electoral fraud can be avoided by allowing each voter to check that his vote has been assigned to the alternative(s) he wants to support. In its simpler version our protocol makes systematic use of the fact that the legitimacy of each voter is first checked and the legitimate voters given the same identification number.

This version, however, cannot be used in elections where the voters are allowed to change their minds about whom to vote. The modification called for is one in which most voters are given different identification numbers. This is done using a protocol devised for contexts in which a party offers a set of secrets for sale to several potentially interested parties. Once this protocol has been completed, each buyer is in the possession of precisely one secret and the seller has no idea which one has been bought by any buyer. In the same way, the voters are offered a large set of identification numbers so that every voter gets exactly one number and the system offering the numbers does not know any voter's identification number.

The fact that several voters may receive the same identification does not hamper the protocol. The voters check whether their votes have been correctly assigned by looking at the values of hash functions devised by the voters themselves. As arguments of the functions feature both the identification number and the vote of the voter. Hence, given that two voters have identical identification numbers, it by no means follows that the values of their hash functions were identical.

Even the more complicated version of the protocol is not sabotage-free. However, as long as each voter who gets an identification number also votes, it is impossible to cheat in this system since any cheating would result in too many votes. The weakness of the above protocols is that only the fact that someone is cheating, not the person who is trying to cheat can be established.

The outlined protocol allows for cancelling and recasting of valid votes. It also allows any voter to produce proof that his vote has not been correctly assigned without jeopardizing ballot-secrecy. These are undoubtedly features not available in current secret balloting systems. On the other hand, the protocol may provide incentives for selling and buying of votes since the contracts are observable in a way that traditional elections are not.

11. References

Benaloh, J.D.C. (1987) Verifiable secret-ballot elections. *Yale University, Computer Science Department, Technical Report 561.*

Chaum, D. (1988) Election with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: *Lecture Notes in Computer Science, vol. 330.* C.G.Gunther (ed.). Berlin-Heidelberg-New York, Springer-Verlag, 177-182.

Diffie, W. and Hellman, M. (1976) New directions in cryptography. *IEEE Transactions on Information Theory,* **IT-22**, 644-654.

Iversen, K.R. (1991) A cryptographic scheme for computerized general elections (extended abstract), *Crypto '91,* Session 9: Secure Computation Protocols.

Nurmi, H. and Salomaa, A. (1991a) A cryptographic approach to the secret ballot. *Behavioral Science,* **36**, 34-40.

Nurmi, H. and Salomaa, A. (1991b) Secret ballot elections and public-key cryptosystems. Submitted for publication.

Nurmi, H., Salomaa, A. and Santean, L. (1991) Secret ballot elections in computer networks. *Computers and Security,* forthcoming.

Rivest, R., Shamir, A. and Adleman, L. (1978) A method for obtaining digital signatures and public-key cryptosystems. *ACM Communications,* **21**, 120-126.

Salomaa, A. (1990) *Public-Key Cryptography.* Springer-Verlag, Berlin-Heidelberg-New York.

Salomaa, A. and Santean, L. (1990) Secret selling of secret with many buyers. *EATCS Bulletin,* **42**, 178-186.