Multiple Administrators for Electronic Voting*

Brandon William DuRette

May 19, 1999

1 Introduction

Electronic voting systems attempt to achieve at least the same level of security as ordinary elections. Voters want to be assured of anonymity; no one should be able to know both the origin and contents of a ballot. Election officials wish to prevent unauthorized persons from voting, and to prevent authorized persons from casting more than one ballot. Finally, the results of the election should be correct and verifiable by any concerned party.

Researchers at the Laboratory for Computer Science had implemented a system called EVOX,† based upon a scheme proposed by Fujioka, Okamoto, and Ohta.[4] [2] While the system has been used at MIT for Undergraduate Association elections, EVOX still possesses certain vulnerabilities. Prior to this undertaking one such weakness was the inordinate amount of power vested in a single administrator server. By distributing the power of administration to several servers and requiring the permission of more than one to cast a vote, a single administrator is prevented from forging a vote.

The rest of this paper describes the problem and the solution in greater detail. Section 2 describes the original EVOX protocol as a basis for discussing the multiple administrator protocol. Section 3 explains the attack on the single administrator protocol we seek to solve. Section 4 further motivates the multiple administrator solution, and suggests possible solutions. The protocol chosen for implementation is discussed in detail in Section 5. Details of the implementation are covered in

*An electronic version of this thesis is available from the Cryptography and Information Security Group at the Massachusetts Institute of Technology.
http://theory.lcs.mit.edu/~cis/theses/DuRette-bachelors.pdf

†Current project status information is available from the Cryptography and Information Security Group at the Massachusetts Institute of Technology.
http://theory.lcs.mit.edu/~cis/voting/voting.html

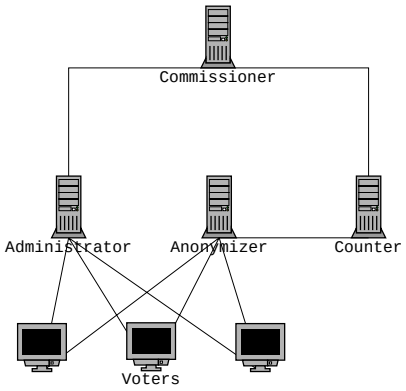

Figure 1: Network diagram of the original EVOX system.

Section 6. Section 7 discusses the project, lessons learned, problems encountered, etc. Section 8 suggests future improvements which could be made to the EVOX system and Section 9 concludes.

2 The EVOX Protocol

The EVOX protocol is based upon the protocol proposed by Fujioka, Okamoto, and Ohta (FOO protocol). The FOO protocol provides provable security, provided that the cryptographic functions that it uses are unbreakable. However, the demands placed on voters to ensure a secure election are impractical in real elections. Specifically, to maintain a secure election, even people who opted not to vote must check to make sure no vote was counted for them.

The EVOX protocol relaxes this requirement. Once a vote is cast, the voter has no further responsibility. However, this introduces potential security weaknesses, such as vote forging discussed in Section 3.

The network diagram showing the servers and connections for the EVOX protocol is shown in Figure 1. The EVOX protocol can be broken down into five stages: preparation, administration, anonymization, collecting, and counting. The details of each of these stages follow.

1. **Preparation**

   (a) Voter contacts an administrator to request a ballot.

   (b) Administrator determines which question lists the voter is allowed to vote on and returns the ballot. The ballot is a signed copy of the questions the voter is voting on.

(c) Voter completes the ballot, commits to it, blinds the commitment, signs the result, and returns the signed blinded commitment of the vote to the administrator for administration.

2. **Administration**

   (a) Administrator checks that the voter has the right to vote. If not, the administration is rejected.

   (b) Administrator checks that the voter has not already submitted a vote. If the voter has already voted, the administration is rejected.

   (c) Administrator checks the voter's signature on the ballot. If the signature is valid, the administrator signs the ballot and returns it to the voter.

3. Voter unblinds the ballot and checks that the administrator's signature is valid. If not, the voter submits a complaint to the commissioner.

4. **Anonymization**

   (a) The voter constructs a message with the administrator signed, unblinded commitment, the commitment keys, and the ballot (the administrator signed list of questions). This message is encrypted with the counter's public key submitted via a secure channel to the anonymizer.

   (b) The anonymizer maintains a store of all the votes submitted throughout the entire election.

   (c) The anonymizer rearranges the order of the ballots randomly to prevent timing attacks. Since the ballots are submitted to the counter in a different order from the order they were received, even if the counter had a log of all submissions to the counter it could not determine the origin of any vote.

5. **Collection**

   (a) After the close of the election, the counter obtains the ballots from the anonymizer.

   For each ballot, the following operations are performed:

   (b) The counter checks that the administrator's signature signs the ballot. If not, a complaint is sent to the commissioner and the vote is not counted.

(c) The counter verifies the commitment for the vote, using the keys attached to the vote. If the commitment does not commit the vote, a complaint is sent to the commissioner and the vote is not counted.

(d) The administrator's signature on the ballot, or question list, is verified. If the signature does not sign the ballot a complaint is sent to the commissioner. The signature is intended to prevent voters from voting in a part of the election they were not authorized to vote in such as Republicans voting in a Democratic primary. However, this signature is not sufficient for this purpose. The protocol needs to be changed to prevent voters from sharing signed question lists with one another (see Section 8.2).

(e) The counter checks that the commitment bytes are unique. If two votes contain the same commitment, it is assumed that they are the same vote and only one copy is counted.

6. **Counting**

(a) Ballots which pass the collection stage are then counted using the opened commitment.

(b) After counting all the ballots, the counter publishes the results of the election and all of the votes which were received. Since the votes are made public, any party that wishes to verify the count of the vote can do so.

3 Vote Forging

In the original EVOX protocol, a single administrator is entrusted with the power to authorize votes. A dishonest administrator can exploit this power to cast fraudulent votes. Because the counter receives anonymous votes, the counter cannot distinguish a vote cast by a legitimate voter and one cast by the administrator. The only way to prove forgery by the administrator is if the number of votes counted in the election exceeds the number of voters registered. Alternatively, if by casting a vote for someone, the administrator precludes them from voting, that voter can complain to the commissioner, but such a complaint cannot be distinguished from a voter desiring to cast a second vote.

An administrator desiring to forge votes can do so with a very small chance of detection. The administrator produces signed ballots during the election, but does not submit them immediately to the anonymizer. At a time very near the end of the election, the administrator determines how many voters have not voted and submits ballots to the anonymizer for those voters. If any of those voters choose to vote in the last few moments of the election, the administrator can either deny the request

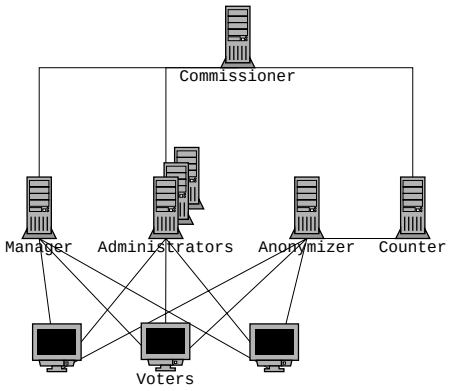

Figure 2: Network diagram of the multiple-administrator EVOX system.

claiming the election is closed, or simply respond (or fail to respond) as if the server is too busy to accept the connection. Alternatively, the administrator could reserve some "safety-factor" votes, by not submitting votes for all of the voters. If a voter votes in the last moments of the election, the administrator behaves normally. The legitimate vote consumes a "safety-factor" vote. Either way, the end result is an election in which the total number of votes is less than or equal to the number of registered voters. Thus, the fraud is undetectable.

4 Multiple Administrators

Distributing the authority and workload between several administrators accomplishes two things. Most importantly, it improves the security of the overall system as a single administrator no longer has the power to forge votes. Additionally, during an election of limited time, a single administrator election is limited in size to the number of votes that the administrator can sign during that time period. By distributing the workload, the system becomes more scalable.

As described above, the EVOX protocol was susceptible to vote forgery by the administrator. In the original protocol by Fujioka et. al. this was prevented by forcing the administrator to publish a list of voters who cast ballots. Voters were responsible for complaining if their vote was not counted, or a vote was forged on their behalf. In practice, this does not work. Expecting people who were unwilling to vote in the original election to check that their vote was *not* cast is unreasonable. Vote forgery must be prevented without additional burden to the voters, and with *no* burden to the non-voters.

To prevent a single administrator from forging a ballot, we simply require more

than one administrator to sign a ballot to validate it. Thus, no single administrator has the power to cast a vote on behalf of a legitimate voter. If the power to sign votes is held by persons having no reason to collude (i.e. opposing political parties or interest groups), it can be assumed that ballot forging by collusion will not happen.

There are two basic methods for enforcing multiple administrators. The most straightforward is to require each administrator to sign the vote independantly. To be valid a vote must then contain several signatures. Another method would be to use a threshold signature protocol.[3] In such protocols, a *single* signature is produced using the keys of several administrators. While threshold signature schemes have a theoretical elegance for this type of problem, implementation of such a protocol was not chosen due to the complexity of the implementation.

A secure implementation of a multiple administrator system is not as easy as choosing one of the two methods above and implementing it. The existence of multiple administrators creates another level of complexity and adds more potential security holes. To illustrate this, here are some protocols which on the surface seem valid, but prove to be either impossible or insecure.

4.1 The simple protocol

The simple protocol is the straightforward implementation of the multiple signature scheme. The system is comprised of n administrators, of which t signatures are required for $t <= n$. In this scheme, double voting is achievable if $t <= n/2$. In that case, there exist disjoint subsets of administrators who could be used to sign different ballots, thereby allowing double voting. The simple solution to this problem is to require $t > n/2$. That way, there are no disjoint subsets of administrators who could sign ballots. What if we do not want to accept this requirement? For instance, if for scalability or any other reason we want to allow any arbitrary t. Can a protocol be implemented which securely allows this? The answer is, maybe. While many solutions have been proposed, none proves to be secure. Some failed attempts at such schemes are presented below.

4.2 Communicating administrators

One might think that voters could be prevented from double voting if the administrators could communicate about what votes they have signed. Administrators would agree to only sign votes that contain the same data as votes signed by the first administrator to sign a particular voter's vote. The fallacy in this method is there is no way for an administrator to know what the vote is that is being signed. That information is blinded from the administrator. The blinding for each administrator

produces different data, so even votes which contain the same data appear different to the administrators.

4.3 Managed administrators

Another possible solution which was considered was the addition of another type of server, the manager, whose signature was also required to cast a vote. The protocol requires that a voter obtain the signatures of $t < n$ administrators as before. To prevent double voting, the voter must also have the manager sign a list of administrators who signed the vote. In theory, the manager would only sign one such list per voter, thereby preventing double voting.

One danger of this protocol is that the manager could collude with voters to cast multiple votes. Worse, until a decent public key infrastructure replaces password authentication, the manager can forge votes without the voter knowing. Once the voter has authenticated himself to the manager for the purposes of attaining a signed admin list, the manager has the voter's password. Using that password, the manager need only construct a new vote and have it signed by administrators which have not already signed for that voter. Once enough signatures are collected, the manager can sign the list of administrators and submit the vote.

No protocol has been discovered which requires fewer than $\lfloor n/2 \rfloor + 1$ administrators. The system which has been implemented is the managed administrator system. Since in this system the number of signatures required is not specified, it is important for users of the system to understand that to be safe, the number of administrators required should exceed half of the available administrators.

5 The Managed Administrator Protocol

The basic idea behind the multiple administrator protocol is similar to the single administrator protocol. The same basic five stages are required, though the stages are not exactly the same. The changes to each stage are as follows:

1. **Preparation**

   The ballot is now obtained from the manager. This change does not significantly change the functionality or security of the system. However, the manager becomes the logical central authority for providing common tasks like distributing ballots.

2. **Administration**

The ballot must now be sent to several a number of administrators for administration. The voter commits and blinds ballots for as many administrators as are required in the election. The ballots that are sent to each administrator contain the same underlying data, but are different because the blinding factors are different. The votes that are sent to each administrator do not have any information about which other administrators the voter intends to have sign the ballot. Only the voter knows which voters signed her vote so the set of signers cannot be used to identify the source of the vote.Each administrator behaves exactly as the single administrator did; one vote is signed per voter. The signatures received from the administrators are copied onto a single vote which will be submitted.

A final step has been added to complete administration. Once all of the signatures have been received, the voter requests a final signature from the manager. The voter blinds the list of administrators who signed the ballot to the manager for signing. The manager will sign one list per voter. This signature is required to prevent voters from validating more than one ballot. The blinding is done to prevent the manager from matching voters to votes based on administrator lists.

3. **Anonymization**

   The anonymization stage is just used to prevent the counter from knowing the origin of the vote. As such, it remains unchanged in the new protocol.

4. **Collection**

   To pass from the collection stage to the counting stage, the ballot must pass two additional tests. First, the manager's signature must sign the list of administrators who have signed the ballot. No omissions are allowed. Second, **all** of the administrator signatures must sign the ballot data. The order in which these tests are applied does not affect the security of the election, but for efficiency the lists are checked before the administrator signatures are. This is done because checking signatures is an expensive operation and the lists could eliminate votes from the counting pool with only a single signature check.

5. **Counting**

   The counting of the ballots is performed in the same manner as the original protocol. Only ballots which pass the collection stage are counted. All votes are published for independant verification. An outstanding security weakness of this protocol is that the counter could refuse to count a vote and modify the signature data before publishing the list. Only the voter could prove that the

vote should have been counted, but because we insist on a single session for the voter, this problem remains. One potential solution is to have multiple counters who could prove a vote was "dropped" by the other counters (see Section 8.3).

6 Implementation Details

The implementation of the new protocol required the addition of a new server, the Manager, as well as changes to several other classes in the system. These significant changes and additions are highlighted in this section. In addition, some other classes were changed in a minor or obvious way. Those changes will not be made explicit in this document.

6.1 Manager

The creation of a new server, the manager, required the addition of several classes. The most important one is the ManagerServer class. This class extends the SecureServer class which abstracts the communication layer out of the server. The major functionality is in the handleConnection method. This method recognizes the two types of messages accepted by the manager and responds accordingly. If the message is a LogonMessage, the manager looks up the question lists for that voter and replies with a QuestionListMessage. The other type of message recognized by the manager is a new message type, the AdminListMessage. This message represents a request for signature from a voter. If the voter has not yet had an administrator list signed, the manager signs the list and returns the signature in an AdminListReply (another new message type). If this is a repeated request, the manager responds with an appropriate ErrorMessage.

In addition to the ManagerServer and aforementioned messages, the manager also requires a ManagerParameters class which stores information which it needs: cryptographic keys, users, question lists, etc. Also, for the purposes of configuring and installing the manager a user interface panel was added to the election builder. This panel is the ManagerSetupPanel and resembles the other server configuration panels.

6.2 Administrator

The new protocol required two changes to the administrators. First, in order to identify an administrator, each administrator now possesses an ID number. This ID number is used throughout the system to match keys to signatures, locate ip addresses for administrators, etc. Secondly, as discussed in Section 5 the administrator no longer

handles the distribution of question lists. LogonMessages received by an administrator are replied to with an ErrorMessage.

As with the manager, the changes to the administrator went beyond the AdminServer class. The installation and configuration utilities have new components to support multiple administrators.

6.3 Election

The Election class is a data structure that stores all the information pertaining to an election. Since an election now has more information, the election had to be modified. Formerly, the election held only one AdminParameters object (used to store keys, addresses, etc. for an administrator). To support multiple administrators, the Election now holds a hashtable of AdminParameters. The table is keyed by the ID number of the administrator. The Election now contains a ManagerParameters object and other information such as the number of signatures required to validate a vote.

6.4 Counter

The Counter has been changed to implement the new collection protocol. It checks incoming votes for a signed administrator list and all of the signatures attached to the vote data.

7 Project Discussion

The multiple administrator EVOX system proved to be an interesting and challenging advanced undergraduate project. The EVOX system itself is a fairly large system, so a great deal of time was spent up front to familiarize myself with most of the different classes in the system and how they interacted. The end of this phase was marked by the UA election, in which the original single administrator EVOX was used. Since then, the focus has been on designing and developing the multiple administrator protocol. The protocol went through two complete revisions, because the first one was unworkable.

The first design was the fundamentally flawed design discussed in Section 4, using a shared data store to determine if the vote had been changed between administrators. The development of that system proceeded without enough thought being given to the blind signature scheme. Neither myself nor the individuals I discussed the project with realized the oversight until late in the development process. While unfortunate, that oversight provided one of the most valuable and widely applicable lessons of the

entire project: spend more time up front to ensure the correctness of a design before proceding with development. It will save you time in the end.

The work done on the first portion of the project was not entirely in vain. Many of the changes were applicable in the second system. The first design provided for an additional server, just as the second system. While the function of that server was different, all of the infrastructure changes for key management and server configuration were transferrable to the final system. Likewise, the infrastructure changes to support multiple administrators such as key management, administrator identification, and configuration were usable in the second generation.

8 Future Work

While this implementation does provide a multiple administrator protocol for EVOX, it should not be considered complete. There are still several places where improvements can be made.

8.1 Public key authentication

Currently the voters authenticate themselves using just a username and password. This is done because no method has been found to integrate a public key verification system into a web browser environment. This password system is inherently weak. Passwords for the users accessible to the administrators with some amount of work. As a result, until a public key infrastructure is in place, there is a risk of administrators forging votes by obtaining users passwords.

8.2 Simultaneous elections

Undergraduate elections at MIT are much like primary elections at the national level. Not every voter is allowed to vote on every ballot item. At MIT, students are designated by their class: freshmen vote for freshmen officers; sophomores for sophomores; etc. At the national level voters are divided by political party: Republicans vote in the Republican primary and Democrats vote in the Democratic primary. EVOX supports this type of election on some level. Users are distinguished at the administrator and are given the correct question lists, signed by the manager. Voters vote in the correct election.

With this scheme, there is a possibility that the manager would be able to give the voter a marked ballot, signed of course, which could be distinguished at the counter. The hope of detecting such a mark dwindles even further if the manager

uses a randomized signature scheme such that even if the ballot data is the same for all voters, the signatures are different.

To improve the situation, perhaps a protocol could be implemented such that the ballots for the election are made publicly available on a webserver. Everyone obtains the ballot from the same place so the ballot data and corresponding signature is the same. This way the manager cannot distinguish votes at the counter, but the problem of preventing people from voting in elections which they are not supposed to vote in reappears. To solve this problem, the manager and administrators need separate keys for separate elections. When the completed ballots are signed, they are signed with the appropriate key for the election which the voter should have voted in. If the voter attempts to vote in the wrong election, the signature, even though it signs the ballot, is considered invalid because it is signed with keys that do not correspond to that election.

A scheme of this sort should be investigated and implemented to further enhance the security of EVOX.

8.3 Vote dropping

Nothing in either EVOX protocol prevents votes from being "dropped on the floor." Either the anonymizer or the counter can successfully drop votes on the floor without much chance of being caught. The danger of the counter dropping votes which it disagrees with is serious. For the anonymizer to fix an election would require speculation about what a voter's likely vote is, based upon limited information (i.e. MIT students vote from 18.x.x.x IP addresses and tend to vote a certain way).

To prevent vote dropping, the anonymizer and the counter should be replicated. This task would be not unlike the replication of the administrators. Each counter would independantly count the election and the results would be compared. If a discrepancy was found, the counter who had the additional vote could prove the vote existed by showing it to the counter which did not have the vote. Discrepancies could be resolved and the vote count would be accurate.

8.4 Anonymizer chain

Currently, only a single anonymizer server is being used. As a result, the anonymizer which submits the ballot to the counter knows the origin of the vote (at least the IP address). Therefore, collusion between the anonymizer and the counter could compromise the anonymity of the voters. Using a chain of anonymizer servers, we could increase the number of colluding parties required to compromise anonymity.

8.5 Database back-end

Many of the servers could benefit from the use of a real database back end. Currently, all information is stored in a regular Unix file system. Using a database would speed up the access to information by various servers. The anonymizer, because it stores all the votes during the election, would be the prime benefactor. Counters could also use databases to store votes during and after counting. Administrators could use the database to store information to authenticate voters (public keys or passwords) as well as the list of who is participating in which election (See Section 8.2).

8.6 New cryptographic protocols

The EVOX system could also benefit from the incorporation of new cryptographic protocols as they become available for practical use. For instance, a lot of theoretical work has been done on batch signature processing, whereby several documents can be digitally signed with the same amount of work as a single ordinary digital signature.[1] If such a protocol could be used in the EVOX system, the load on administrators signing votes would be greatly reduced.

9 Conclusion

This project sought to improve the security of the EVOX electronic voting system by distributing the power of administration to several parties. The goal of multiple administrators has been accomplished. Further, in the process of working on the multiple administrator system, several other areas of the EVOX system have been scrutinized and ideas for future work have been conceived.

References

[1] Amos Fiat. Batch RSA. *Journal of Cryptology*, 10(2):75–88, Spring 1997.

[2] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology—AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251, Gold Coast, Queensland, Australia, 13–16 December 1992. Springer-Verlag.

[3] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Robust threshold DSS signatures. In Ueli Maurer, editor, *Advances in Cryptology—*

EUROCRYPT 96, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371. Springer-Verlag, 12–16 May 1996.

[4] Mark Herschberg. Secure electornic voting using the world wide web. Master's thesis, Massachusetts Institute of Technology, June 1997.