The SAVE system — secure architecture for voting electronically

T Selker and J Goler

Existing technology is capable of yielding secure, reliable, and auditable voting systems. This system outlines an architecture for polling place electronic voting, based on redundancy at each stage of the ballot submission process that is resistant to external hacking and internal insertion of malicious code. The proposed architecture addresses all layers of the system beyond the point when a voter commits the ballot. These steps include the verification of eligibility to vote, authentication, and aggregation of the vote. A redundant electronic audit trail keeps track of all of the votes and messages received. There is no single point of failure in the system, as none of the components at a particular layer relies on any of the others; nor is there a single component that decides what tally is correct. Each system arrives at the result on its own.

1. Introduction

Computation systems are designed to be the most reliable systems for tabulation. By their very character, they are not subject to the kinds of mechanical failures that plague traditional voting equipment.

Despite the advantages electronic systems offer, several papers and well-known authors [1] have raised fears, uncertainties and doubts as to the effectiveness and trustworthiness of electronic voting equipment.

Electronic systems have several benefits over paper systems, specifically they remove errors in collection and tabulation of ballots, speed the tabulation process and reduce the cost and overhead in acquiring and handling paper supplies. In addition, if properly designed, electronic voting systems can actually be more inherently secure than paper-based systems.

computation systems are designed to be the most reliable systems for tabulation

It is possible to create electronic voting systems that, by their very nature, are secure, reliable and trustworthy. An analysis of types of possible attacks, the possible scope of these attacks and the likelihood that they will occur is a place to begin. The architecture should address these vulnerabilities.

This paper will demonstrate an approach for using existing technologies in the form of computers and their networks to effectively and efficiently handle the voting process. Indeed, the proposed approach would solve current problems while improving efficiency.

electronic systems have several benefits over paper systems

Specifically this paper will lay out an *n*-version [2] type of voting system, comprised of multiple, independently developed modules, that addresses the issues of:

- accurate transmission and recording of voter intent, resulting from an architecture that performs fault detection and correction,
- prevention of outside tampering or hacking, especially involving the threat of changing votes,
- prevention of malicious internal fraud involving changing or specifically developing malicious voting system components,
- interception of vote transmission or falsifying the contents of messages between system components.

Voting is a complex procedure. This particular paper will not address the important difficulties of registration, local record messages between system components. If the system is not over an open network, this threat is of far less concern.

2.1.3 Malicious voters

A voter gaining access to the system could try to vote more than once, or as another person, or try to steal the votes of other individuals. While to date care has been taken to limit access to smart cards or other methods to opening a poll, it is possible and important to improve access control to the voting act. Coercion is always a danger; technology can be used to allow or to reduce coercion as well.

2.1.4 Malicious election administrators

An unscrupulous election administrator could activate additional ballots, vote as many people, alter the counts and attempt to destroy or alter ballot images. By using multiple component keys and distributed architectures, it would be more difficult for a single administrator or even a small group to compromise an election.

it should be assumed that a hacker could discover the source code through some means

2.2 Security

Typical methods of implementing voting security focus on:

- isolating the process so that no one can see or change a vote,
- building in review.

Typical governmental applications have relied on isolation and confidentiality as a security approach, labelled by some as 'security by obscurity'. The most modern conversations about security describe the value in oversight either by expert review, redundancy or open-source methods.

In the first case, confidentiality as a security approach has worked well. Potential hackers have not had access to the software and have not known what to do when they have access to it. Software systems for voting are relatively new. People attempting to compromise security in elections have not been sophisticated hackers yet. The approach of prohibiting access to things that should remain secure has been very successful. A final key point is that voting conditions change with time, many ballots being finalised within a day of the election. The concerns that would alter a specific ballot tend to be more local and time dependent than concerns about trying to bring down a country or economy.

Certainly secrecy itself is a key method of preventing the wrong people from gaining access to sensitive data. However, secret and closed systems present the serious problems of Easter eggs and backdoor approaches. While these problems may seem far-fetched, they have to be taken seriously, because it is possible that a set of voting machines in a particular precinct could be turned into zombies by setting them to a testing mode.

Such tampering, of course, would be easily uncovered due to the discrepancy with the number of registered voters casting ballots. However, if, for instance, four to seven officials at a balloting place agreed to work together, they could cast ballots after the voters left. While these methods seem to have worked for a long time, there have been breaches of security in many elections. Most have been isolated incidents and have had little impact on the national level. Thus they have merited little national scrutiny, however after the 2000 US election, all errors have been made highly public.

However, with the prospect of large-scale, undetectable fraud by using a single system, it becomes more important to have an *n*-version system, with full auditing along the process.

In addition, security means protecting the voting system's ability to operate effectively for the entire period of voting. Thus, it must be able to resist denial of service attacks, malicious physical attack and loss of electricity. Much of this security derives from the effective operation of the election and not just the equipment.

3. Architecture overview

Designing secure systems requires attention to many levels. Our approach begins by ensuring that there is no single point of failure after the ballot leaves the eyes of the voter. The security starts with the general system concept and goes down to specific ways that the code is written to avoid introducing reliability problems at any stage. The key advantage of this *n*-version architecture (Fig 1) is that structurally there is no way the whole system can be compromised without compromising a very significant number of the parts.

The principle of redundancy is central. It enables the system to continue to work even if there is a failure somewhere along the line. Having multiple programs that process each stage of the ballot casting can establish improved reliability; regardless of how they are written, regardless of who has written them, and regardless of whether they are the same code. Because these versions can be transmitting over different networks, the system is more reliable. Because these are different programs, subverting one of them would not affect the others and still would ultimately enable an accurate vote to be cast. More importantly, if different people and organisations write these modules, intentional tampering of one module (discussed as the 'evil equipment developer' in section 2.1.1), such as putting in an Easter egg (a secret module of code that invokes undocumented functionality), would not affect the integrity of other modules.

However, to be sure these new measures are effective; the system will have to be tested beforehand. By forcing each module to comply with the abstraction-function behaviour that we specify, the architecture will be uniformly black-box testable. In addition, there must be no difference between a test vote and a real vote, as far as the software is concerned.

In our system we separate the aspect of user interface from the rest of the voting system. The intent is to allow user-

4.1 *The user interface*
Perhaps the most vital component of any voting architecture is the user interface. This architecture allows for the user-interface modules to be developed independently of the rest of the architecture. This flexibility permits faster progress incorporating human factors' research in improving the voting experience. Other work [8] establishes user-interface quality assessment. This architecture recommends that all available effort be put into building a user interface that is extremely effective for efficient and accurate voting.

The user interface takes two inputs — the interface definition and the blank ballot. Both of these components are XML documents. The interface definition describes the way in which the UI is to render a ballot.

The user interface collects the votes of the user, as well as the registration data. It then encrypts the ballot using keys from the aggregators. The registration information is added to the encrypted ballots, and the resulting packages are then transmitted to the registration system.

When the user approves the ballot, there will be an *n*-version type system of digital cameras mounted to the DRE that can take a picture of the ballot, or redundant device drivers that observe the actual ballot on the screen and record the contents. To prevent the production of an actual receipt, the picture can only include the ballot itself, and no other features, so that either the ballot is showing entirely or the ballot is obfuscated, so a user cannot put his or her face in the way, or put a piece of paper saying 'Alice Bobster' in the way. Nevertheless, since the digital photograph back-up is not used as the primary counting mechanism, this problem is of little concern for coercion and vote buying.

4.2 *The registration system*
The registration system is the centre of this voting architecture. The registration server has access to the roster of all registered voters. When the registration receives a ballot package containing registration information and an encrypted ballot, it looks at the database, checks to see if the user is valid, and then makes an entry in the database checking off the user as having sent a vote to the aggregator.

Each registration module extracts the encrypted ballot, signs it, and then sends it to the witness modules (see section 4.3) for their signatures. Once the witnesses return their signatures, the signatures can be appended to the encrypted ballot. Then the whole ballot package (without individual identifying information) is shipped off to the aggregators.

4.3 *The witness module*
The witnesses are the simplest of the modules. They take as input an encrypted ballot and produce a signature. Signatures are produced using MD5/RSA [9]. The ballot is digested, and a hash is produced, which when combined with the witness's private key, produces a number that, as far as we know, can only be produced by the holder of the private key. Witnesses do not maintain a record of the ballots coming through them, as they are meant to be lightweight implementations, preferably using separate databases or smart cards so they can be handled easily. Witness modules are to be provided by independent organisations (e.g. political parties, watchdog organisations).

4.4 *The aggregator module*
The aggregator module takes encrypted ballot packages as input. The packages contain the encrypted ballot and a series of signatures produced by the registration system and witnesses. The aggregator parses the signatures and uses the witness public keys to verify the signatures. The aggregator then determines that a set threshold of signatures verify and then decrypt the ballot. Once the ballot is in plain text, the selections are parsed and recorded. Both the encrypted and plain text versions of the ballot will also be stored in a repository.

4.5 *Messaging protocol*
The messaging protocol is based on XML. Communication between modules is simple. The listening module waits for connections; the signalling module then initiates a socket connection, opens an output stream, then an input stream, and writes a string containing the command to the listening module. The module then does its processing and writes a string of commands indicating its response. The output stream is closed first, and then the input stream is closed. Standard sockets are used to connect between various components. The prototype implementation uses the Java Socket and ServerSocket classes that are conveniently provided by JDK 1.4.

5. Security and reliability via architecture
The architecture of this system uses modularity and threshold agreement for fault and hack tolerance. Redundant audit trails enforce certain security and reliability. Modularity is an important cornerstone of any system that can be scrutinised. Each component we have developed is a few hundred lines at the most. And most of that is simply placing the data into a database. The tightness of the software code allows it to be quickly and easily certified to do what it is intended to do, for its compliance with the protocol that demands plug-and-play interaction with the rest of the architecture, as well as code that is easily viewed by outside agencies to determine its accuracy and correctness. Additionally, the separation into interoperable modules creates a voting system that could be modified in one aspect without affecting the certification of another aspect or component. This modularisation dramatically lowers the cost in time and money for certification as systems are created and improved. The most important part of modularity, however, is that by separating the modules by steps we can analyse security in each stage.

Each module keeps track of the other modules it is supposed to send and receive information from, as well as the public keys of those servers. Modules are defined by a contract that indicates what they are to send, receive, and process. By creating a standard contract, anyone can write to the standard and plug a module into the working environment. The architecture itself enforces security and reliability while improving maintainability.

In the previous stage there are *n* systems, each of which provides a piece of data. We cryptographically verify the data for each of them, checking their keys and signatures and voting system architecture outlined in this paper takes the distributed approach to security to another level. Instead of relying on a single company to provide a system in a region, we are relying on the distribution of people to avoid fraud. The architecture becomes more secure when more people are involved. In some cases, too many people involved produce sloppy and buggy code; however, by the very architecture involved, this system becomes more secure and reliable as additional modules are added. Even if some of the organisations have their own political agenda (and act on it), the architecture will maintain the integrity of the system.

Multiple groups create versions of the same part of the architecture. It must be easy for an election administrator to pick *n* of these systems, and run them seamlessly. Thus, there should be a common registry of these modules, and an effective means of ensuring integrity.

6. Conclusions and future work

This voting architecture provides a means to vote over open networks in a way that is reliable, secure, and private. Due to its modularity and common specifications, it is easy to implement, improve and it is inexpensive. The system uses COTS equipment for the all of the back-end systems, reducing the likelihood of fraud with the system components as well as keeping the cost down. These innovations make it particularly attractive for implementation as state budgets are increasingly tightened.

N-version programming can be a powerful tool for improving electronic voting security. The next steps to creating an *n*-version programming voting system are using it to secure the user interface and using it to secure back-end vote tabulation and storing.

Much work remains to be done in the voting architecture field. Our group is working on developing effective user interfaces and improved registration systems. We are also examining ways of providing verifiable feedback to users, but in a way that does not compromise the confidentiality and receipt-freeness requirements of voting. To address the need for clear, balanced ballot forms, we are developing an artificial-intelligence-based system to help inform ballot designers.

Acknowledgements

We would like to thank the John S and James L Knight Foundation and the Carnegie Corporation of New York for their financial support of our research. In addition, we thank the members of the MIT-Caltech voting technology project for their advice. We thank Soyini Liburd for her work on further developing, and analysing an *n*-version voting system.

References

1 Liburd S: 'An N-Version Electronic Voting System', (July 2004).

2 Saltman R: 'Adopting computerised voting in developing countries: comparisons with the US experience', CPSR Newsletter, Computer Professionals for Social Responsibility, 16, No 1, pp 13—16, Computer Professionals for Social Responsibility (Winter 1998).

3 Fujioka O O: 'A practical secret voting scheme for large scale elections', AUSCRYPT 92, pp 244—251 (1992).

4 Avizienis A: 'The methodology of N-version programming', in Lyu M R (Ed) 'Software fault tolerance', Chapter 2, pp 24—46 Wiley (1995) — http://citeseer.nj.nec.com/avizienis95methodology.html

5 Kohno T, Stubblefield A, Rubin A and Wallach D: 'Analysis of an electronic voting system', (July 2003).

6 United States General Accounting Office: 'Elections: Perspectives on Activities and Challenges Across the Nation', (October 2001).

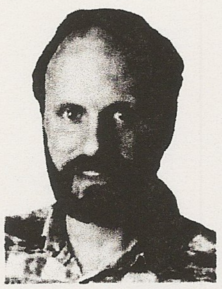
7 Fischer E: 'Voting technologies in the United States: overview and issues for congress', (March 2001).

8 Selker T: 'User interface and ballot design as part of an improved voting system', (May 2001).

9 Rivest R: 'Security in Voting Technology', House Testimony (May 2001) — http://theory.lcs.mit.edu/~rivest/rivest-may-24-01-testimony.txt)

10 Rivest R: 'The MD5 Message-Digest Algorithm', IETF RFC 1321 (1992).

11 Caltech: 'Voting: what is, what could be', MIT Voting Technology Project (July 2001).



Ted Selker is the MIT director of the Caltech/MIT voting project, which evaluates the impact of technology on the election process. A large part of his work in voting is concerned with inventing and testing new technologies for voting. Examples include new approaches to user interface and ballot design, and secure electronic architectures and approaches for improving registration.

His Context-Aware Computing group at the Media Lab strives to create a world in which people's desires and intentions guide computers to help them. This work creates environments that use sensors and artificial intelligence to create keyboard-less computer scenarios.

Prior to MIT, he was an IBM Fellow and directed the User Systems Ergonomics Research lab at IBM. He has served as a consulting professor at Stanford University, taught at Hampshire College and Brown University, and has worked at Xerox PARC and Atari Research.

Jonathan Goler received his BS and MEng from MIT. He has been actively involved in the MIT/Caltech Voting Technology project since its inception, and developed the first n-version voting architecture. He is also active on the IEEE P1583 electronic voting equipment standards committee.

In addition, he serves as a researcher in the Synthetic Biology Working Group at the MIT AI Lab, where he develops design and simulation tools for synthetic biological systems.