Secure Voting Using Disconnected, Distributed Polling Devices

David Clausen, Daryl Puryear, Adrian Rodriguez
Stanford University, Department of Computer Science

June 5, 2000

Abstract: We present a system for secure electronic voting which does not rely on persistent network connections between the polling places and the vote tallying server.

1. Introduction

1.1 Background and motivation

Electronic voting has been discussed extensively in the academic literature, and in some places trials are being conducted as a precursor to implementing a large scale system. All of the systems of which we are aware require that each polling place be connected via a persistent wide area network link to at least one central vote tallying server (essentially a transactional database system which implements some sort of cryptographic protocol). These systems have a significant weakness in that if the network link is severed, or if the central database server crashes or becomes unavailable for some other reason, the affected polling place(s) are rendered useless until the situation is resolved. The implications of this dependence are significant when one considers the time critical nature of many election scenarios.

In the United States, polling places are required by law to accept votes only during a restricted set of hours on one particular day. A network outage which caused polling places to be closed for even a few hours could have devastating results. Imagine a case where a saboteur from the conservative East side of town climbs under a bridge and cuts the exposed fiber−optic trunk lines serving the liberal West side of town, completely cutting off all data and voice service to millions of registered Democrats just as they are coming home from work on election day. Inevitably the local election results would have to be invalidated, but even worse, the sabotage could conceivably "swing" the state's electoral college votes from one party to another, possibly invalidating the national presidential election. Placing such power in the hands of every telephone company technician is inconceivable, and yet this is an underlying assumption of any real world electronic voting system which relies on well connected polling places.

1.2 Our contribution

Our contribution is to repair this deficiency in existing systems by modifying them to work in a disconnected (or more accurately "intermittently connected") environment. We borrow ideas from research on disconnected filesystems, public key cryptography, digital signatures, and newly available low cost cryptographic hardware devices such as java−powered iButtons and smart cards. By mixing these together in an intelligent way along with existing cryptographic voting protocols, we have created a new system with most of the desirable properties of other designs, but which behaves well in the absence of network connectivity.

Additionally, by implementing a prototype of this system, we have demonstrated its technical feasibility, and we have established a new data point for practical systems which seek to combine strong multi−factored user authentication, ballot security and privacy, ease of use, high availability, scalability, low cost, integration with existing voter registration procedures, and compatibility with existing laws.

2. System design

2.1 Overview

The operation of our system is best described by walking through the complete process of preparation, registration, voting, and tallying of votes.

Figure 1. iButton with colored key fobs, and the interface device (reader)

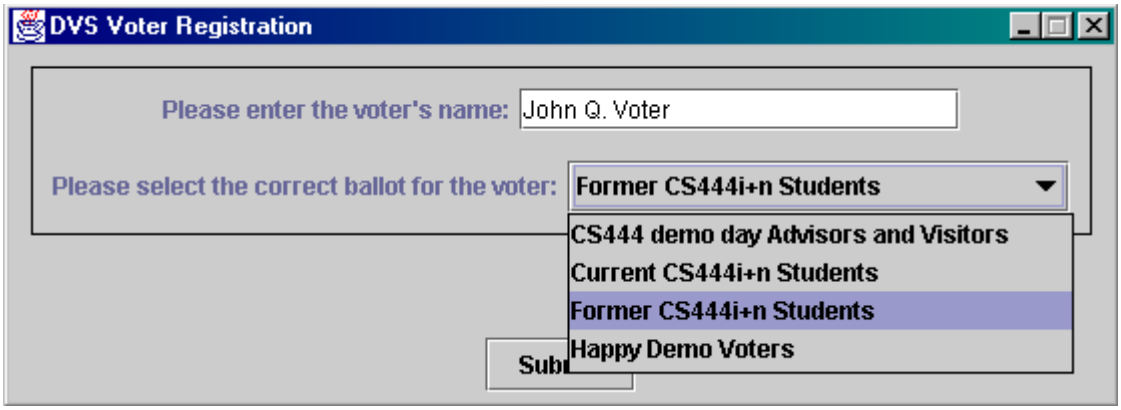

Figure 2. A simple registration interface

2.2 Preparation

The first stage in the process is for a trusted authority to generate a public/private key pair and store these on an inexpensive, tamper−resistant, personal cryptographic device. Ultimately these keys will be used by the voter for authentication and for generating a digital signature on his completed electronic ballot. Once transferred onto the device, the private key will never be revealed, and it is important that the authority which generates it does not keep a copy. The public key must also be signed by the authority so that in the future, one can verify that these are in fact the keys which were anonymously generated by the true authority.

In our prototype, we used a Java−powered iButton as the device, and the keys were 512 bit integers designed for use with RSA encryption. A smart card or other similar device could just as easily have been used, as could a different scheme for digital signatures, such as DSA. The requirements for the device are that it have a modest amount of persistent memory (~2kB), be tamper−resistant, and be able to execute small compiled programs which may perform modular arithmetic on large integers.

At this stage, the devices are not yet linked to any particular voter (or any particular voting district, for that matter), and so we recommend that the key generation process be completed in a bulk fashion by some party which is unrelated to the voting district which will ultimately distribute the buttons to the voter. The devices can then be shipped in large quantities to the voting district authority pre−loaded with keys. Upon receipt, the district administrators can verify the authenticity of the keys by checking the signature on the public key.

2.3 Registration

At some point, a voter who wishes to participate will fill out a paper form containing his name, address, and a handwritten signature, and mail this to his voting district authority. Alternatively the district administrators could consult their records of previous registration forms and proceed similarly. The form is scanned using a conventional desktop optical scanner, and an image of the voter's signature is stored in a database, along with his personal information (name, address, etc.), a unique voter identification number, an indication of which ballot he is to

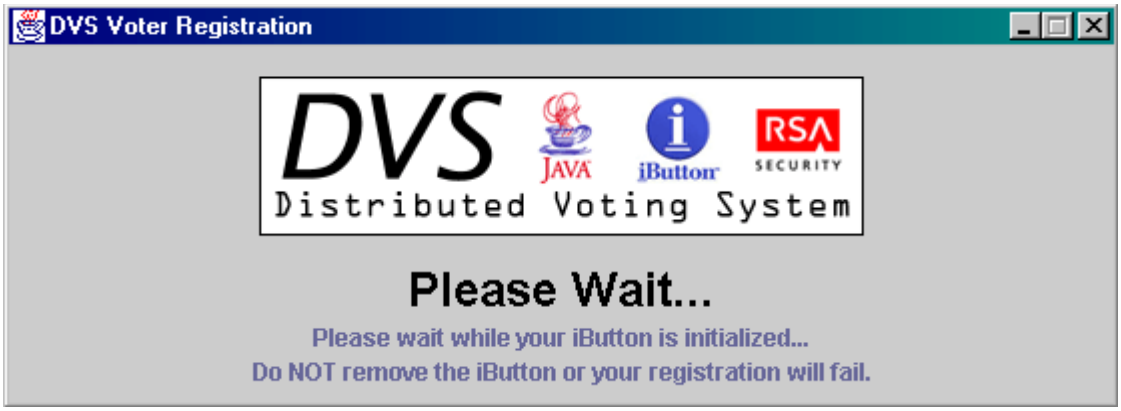

Figure 3. Programming the iButton takes approximately 10 seconds

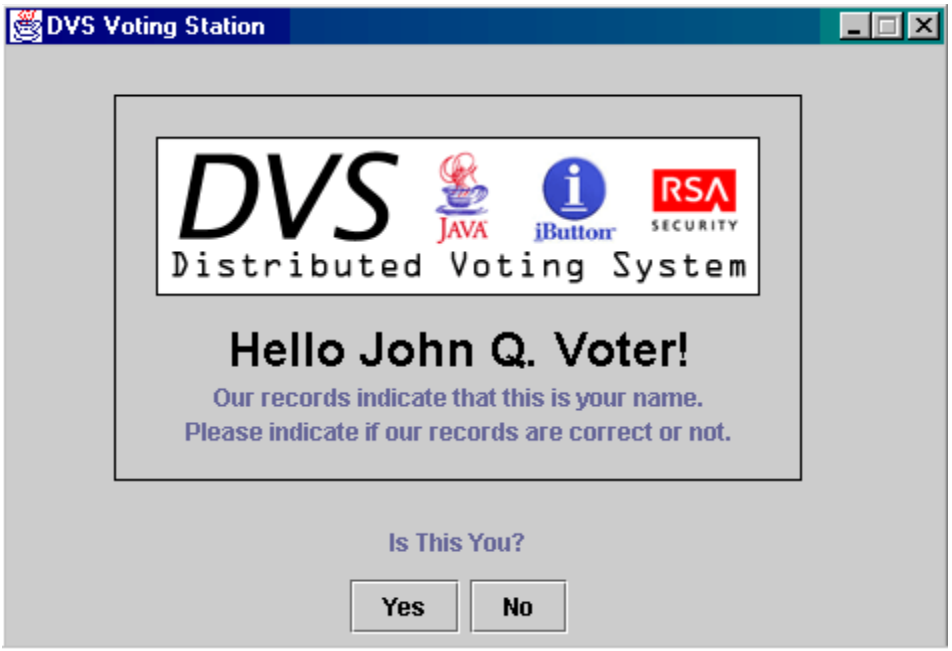

Figure 4. The polling computer reads data from the iButton

be presented (in the case where voters receive different ballots depending on where they live in the district), and a copy of the public key taken from an iButton.

The iButton itself is then programmed with at least the voter's name, identification number, ballot number, and district number. These items, along with the public key, are then signed using the district authority's private key, and this signature is also stored on the iButton. It is then mailed to the voter's home address. along with instructions on where and how to vote.

2.4 Voting

Prior to election day, the voting district administrators establish one or more polling places in the community. A polling place consists of one or more computers, isolated by physical barriers to prevent others from seeing the screen. Each computer is loaded with the appropriate software and keys for interacting with the voters and encrypting the ballots. Each computer also contains a list of the available ballots and associated ballot id numbers for the district(s) it will be serving. The computers each have an iButton reader as well as a touch screen or an external tablet device capable of recording a handwritten signature. If there are multiple computers at a single location, then it is desirable that they be connected via an ethernet or other LAN. Ideally at least one of the computers would have a modem or other wide area network interface for communicating with the master vote tallying server.

On election day, the voter takes his iButton to any one of the polling places administered by his district (or perhaps to one administered by a neighboring district which has a "roaming" agreement with the voter's home district). He inserts his iButton into the receptacle at one of the computers, and the computer reads his name, district id, voter id, ballot number, and the district's signature on these items.

If the signature verifies, the computer presents the voter with the appropriate ballot. Using the mouse or a touch screen, the voter marks his choices on the ballot and clicks the "submit" button. The computer confirms that valid selections have been made, and if so, requests that the voter sign his name to the touch screen or attached tablet device. After the signature has been completed, the computer performs an encryption on the marked ballot using one of the encryption schemes described in section 3.2. This encrypted ballot, along with the image data of the handwritten signature and a timestamp, is sent to the iButton, which returns a digital signature on the data.

This information is then signed by the computer using its private key, and then everything is saved to a file on the hard disk (and possibly replicated onto multiple partitions). Once saved, the iButton is sent a final command indicating that a vote has been cast, which will put it into a "blocked" mode for the rest of the day, preventing the voter from returning to the polls and casting another vote. The voter is informed that the voting process is complete, and he removes his iButton and leaves.

The file which was just saved to disk then begins its journey towards the master tallying server. Under ideal conditions, the file would be instantly transmitted to the tallying server over a directly connected WAN link, at

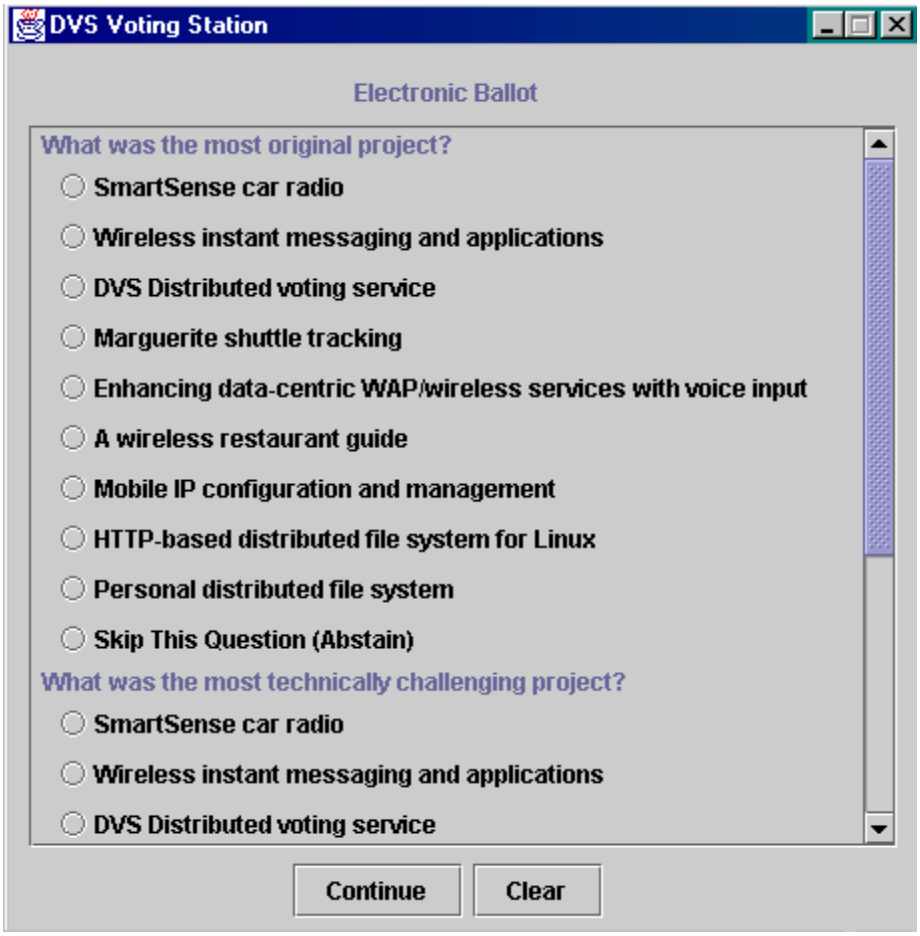

Figure 5. A sample ballot as presented to the voter

which point the server would respond with a "committed" message, and the file could be deleted from the local disk. Direct connection may not be available, however, and so the local computer should replicate this file with some number of its peers (to protect against disk failure), and/or propagate the file to some intermediate relay server which is presumably closer to the final tallying server than the local host. If no connections whatsoever are available, the local computer can simply hold onto the file and wait until another host is available for replication.

In our prototype, we simply had a background process on the polling computer which woke up several times a minute, checked a particular directory for completed votes, and if it found any, attempted to contact the tallying server via an ethernet link. If the server was reachable, the votes in the queue were transmitted and moved to a "sent" directory. Otherwise the process went back to sleep.

At some point the master server will receive the file and issue a "committed" message which may follow an equally arbitrary path back to the local computer, at which point the local file may be deleted from the disk.

2.5 Tallying

When all of the polls have closed, the tallying server must continue to collect votes until it has confirmed that it has received each vote collected by each polling computer. Once this is complete, the tallying server can begin the process of validating and counting votes. Validation consists of verifying all of the digital signatures contained in the encrypted vote file, eliminating identical duplicates which arose from multipath data transfer between the polling and tallying computers, and comparing the handwritten signatures captured by the polling computers with those scanned from registration forms. Although software exists for comparing handwritten signatures, it is likely that human supervision would be required during this process (in our prototype, we simply presented the two images side by side and asked the operator to decide if the signatures matched).

Once validated, votes are then tallied using one of the protocols discussed in section 3.2, and the totals are published.

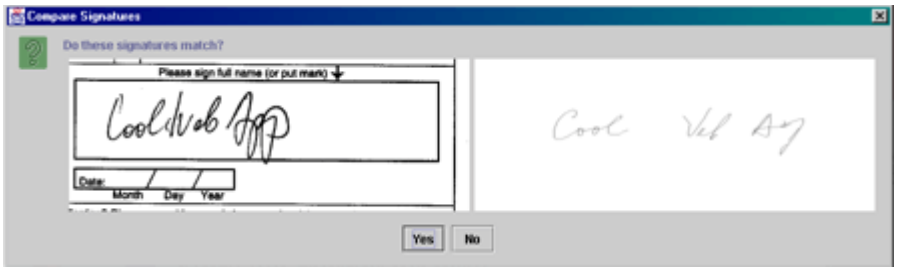

Figure 6. Comparing handwritten signatures

3 Security Analysis

3.1 Overview

To look at the security of the election system, we will consider three different aspects: the security of actual votes (both authenticity and secrecy), preventing the loss of valid votes, and preventing the addition of fraudulent votes. Finally, we will describe potential ways the overall security could be compromised and consider the difficulty of doing so.

3.2 Protecting individual votes

Individual votes must be protected in two ways. First, the authenticity of the vote must be preserved so that the choices of the voter are counted in the election. No one should be able to change the contents of the vote once it has been submitted. Second, the secrecy of the contents must also be maintained so no third party could verifiably ascertain how an individual voted. This is necessary to prevent vote buying and other types of coercion that might affect the outcome of the election.

As described previously, voting is performed on a trusted tamper−proof voting device that interacts with a user−specific authentication device. Once the voter has entered his/her choices, they are formatted appropriately and encrypted on either the voting device or the authentication device. The choice of where this encryption is performed is a trade−off between trusting the tamper−proof nature of each device, along with the logistical requirement that encryption on the extremely low−cost, low−power authentication device could take a considerable amount of time. For our purposes, we are performing the encryption on the voting device. This implies a certain amount of trust that the voting device is not altering the choices entered by the voter in any way before encrypting them. One potential advantage of performing the encryption on the authentication device is if it has the ability to display or indicate the selected choices for confirmation by the voter before encrypting. More about the specific encryption method will be covered in the section on vote secrecy. After the vote is encrypted, it is paired with the digitally captured handwritten signature and digitally signed using the private key on the authentication device. The resulting signature and encrypted information is then ready to be synchronized back to the server − the digital signature will prevent any undetected tampering with the contents.

The other critical aspect of protecting an individual vote is keeping the contents of individual votes secret. This is accomplished through the use of advanced cryptographic protocols. For this project, we simply reviewed existing protocols and built a framework where there are multiple options that can be used to provide cryptographic secrecy. All of the protocols we reviewed were designed to provide desirable cryptographic properties in a connected environment where the voter and server could directly communicate in some form or another during the voting process. However, with minor adaptations, some of these protocols can be adapted to a disconnected environment. Two protocol options are one proposed by Sako and Kilian and another by Cramer, Gennaro, and Schoenmakers.

The cryptographic voting protocol proposed by Sako and Kilian provides a couple of useful properties. First, it provides a receipt−free property that guarantees the secrecy of the vote. Second, it provides universal verifiability so any independent observer can verify that the votes were tallied correctly. To provide the receipt−free property, the protocol requires a physically untappable channel from the server. This can be achieved in our disconnected case by seeding the trusted voting device with a shared secret between it and the server and using the device to generate the necessary channel information. To provide universal verifiability, the votes (without authentication information) are sent through a secure mix−channel that permutes the ordering so the ordering cannot be traced back to the original order of submission to the mix−channel. Then the votes can be decrypted and tallied.

Cramer, Gennaro, and Schoenmakers propose a system where each voter simply has to generate and post their vote along with a proof of its validity to a public forum. They utilize a homomorphic encryption scheme so the encrypted votes can be combined into a single encrypted total that is then decrypted. Under this protocol, the vote never has to be separated from the authentication information because

the individual vote is never decrypted. To ensure this, the decryption key can be distributed among any number of different parties which all have to agree to perform a decryption. If the key is divided up into a significant number of pieces and distributed to a range of groups, it would require collusion among an unreasonably large number of people to decrypt a single vote. This protocol can also be extended using deniable encryption to provide the receipt−free property.

3.3 Preventing loss of valid votes

A new method of corruption introduced by disconnected voting is the ability to prevent votes from propagating back to the server. As part of our protocol, we need to make sure, with reasonable certainty, that all votes cast and accepted by voting devices are subsequently propagated back to the server. One possible cause of loss is accidental through hardware failure. To prevent this from losing votes, votes should be replicated among peer devices whenever possible. Since the signed and encrypted votes will be made public on the server anyway, extra peer sharing of votes is not a security risk. Another possible cause of loss is if a malicious machine is able to sit between the device and the server and not forward some or all of the votes it receives from the device on to the server. To prevent this from losing votes, voting devices should sign the entire submitted vote with their private key and pass the signed version of the vote along to the server. Then the device will not delete the vote from local storage until it receives a signed confirmation from the server. Finally, before performing the final tally, the server must confirm with all clients that all votes have been submitted and confirmed or else votes could still be missing. Together, these mechanisms should prevent both accidental and malicious loss of votes.

3.4 Preventing addition of fraudulent votes

The last important element of the system security is to prevent the addition of fraudulent votes. Since all votes have to be validated against entries stored in the voter registration database, any fraudulent votes cast as non−registered voters will clearly not be validated. This does not, however, deal with the case where a vote is submitted in place of a registered voter who would otherwise not vote for some reason. Since the voter is sent an authentication device that is used to digitally sign his/her encrypted vote, the fraudulent voter must first steal the authentication device from the actual voter (or from the mail in transit to the voter). To prevent mail fraud from being sufficient to cast a false vote, we also require a handwritten signature when the vote is cast. The handwritten signature is then digitized and sent along with the vote to be compared against the handwritten signature on file in the registration database. Since the current system is based on authentication by signature, this provides the same level of protection against fraudulent voting as the current system. Besides forging a signature, the fraudulent voter must steal the authentication device so there are additional risks involved in submitting a fraudulent vote.

Since voters are now able to vote from any voting device in a disconnected manner, it is also important to prevent duplicate votes. Since the server can store the votes together with the authentication information, the server can easily detect duplicate votes and then it is up to the election officials how to determine which vote is counted. A likely solution is to examine the timestamps and accept the first vote cast.

3.5 Security failure models

With the addition of digital signatures, it becomes much more difficult for fraudulent votes to be cast. However, with the addition of public keys in the registration database, if the contents of the registration database could be altered, then the hacker could easily submit a fraudulent vote which passed the digital signature verification since they would have generated both of the necessary keys. This can be made extremely difficult, however, since the registration database does not need any outside Internet connections for voter lookups. All that is necessary is a connection to the voting server when votes are being validated, but this could be handled with a local physical connection. This removes electronic hacking as a method for corrupting the registration database and the use of strict physical security measures could be applied to protect the database against unauthorized modifications.

4. Feasibility Analysis

4.1 Overview

Security is not the only property of a voting system which will affect its adoption by government agencies. Scalability, availability, fault tolerance, cost, ease of use (by voters and by administrators), and conformance to legal standards are also important.

4.2 Scalability, availability, and fault tolerance

As discussed in the introduction, our voting system provides superior availability and fault tolerance when compared to fully connected designs. This comes from its distributed nature, which permits nodes to take advantage of whatever network connectivity may be available whenever it is available.

The distributed nature of the system also improves its scalability. Polling devices can be added, relocated, and replaced as needed to meet demand. Each new device needs to be given its own key pair, and the tallying server needs to store the public key in a database, but this is a minor incremental burden.

Similarly tallying servers can be added as necessary, since vote collection and most of the verification process can be completed in parallel by multiple servers. On the other

hand, if an insufficient number of servers are available to accept all incoming connections, the polling computers will simply enqueue their completed ballots and attempt to connect later.

4.3 Cost

Any electronic voting system will require the use of computers at polling places and for vote tallying. Our system imposes the additional cost of one iButton (or similar device) for each participating voter. The iButtons we used in our prototype have a retail cost of $12 each in small quantities. Estimates we have seen of the cost per voter for current elections in the United States are roughly around the $10 mark, so it would seem that we are at least within an order of magnitude of an acceptable cost.

When compared to other electronic voting systems, the cost of the iButtons in our system can be offset by the reduced administration costs and direct expense associated with establishing persistent network connections between the polling places and the tallying server. For example, a fully connected system might require the purchase of a bank of modems and associated phone lines at the tallying location in order to provide reasonably reliable connectivity. Whereas in our system, communication could occur over the internet, and polling places could use a standard dialup service such as AOL.

4.4 Ease of use

Computerized voting systems present a problem for people who aren't familiar with computers, or who are unable to use standard I/O devices due to visual impairment, lack of dexterity, etc. Our system does not really address these issues directly, but we have tried to make very few assumptions about the user interface in order to leave options open for dealing with these kinds of problems.

The one important requirement we impose is that every voter must have a properly programmed iButton with him at the polling place in order to vote. This is necessary to identify the voter, to decide which ballot to present, and to generate the digital signature on the completed ballot. Although small, the iButton has a rather awkward form factor, and since it has no use to the voter except on election day, it would be easy to loose or forget about. This may present a significant usability problem. A smartcard may be more appropriate for this task, since its shape would be more familiar and convenient for most voters.

4.4 Legal conformance

Legal restrictions on voting may be a significant deterrent to real world implementation of this or other electronic voting systems, at least in the United States. In particular, a number of restrictions were put into place during the civil rights movement of the 1960's which were designed to protect against improper disqualification of voters, and among these are explicit prohibitions against requiring voters to present any form of identification when voting.

While clearly our design violates this prohibition, we believe this may not be fatal if the electronic system is implemented in parallel with the existing system, similar to the procedure for absentee ballots. The current system would be left intact, so voters could cast their vote at their designated precinct polling place without presenting identification, or they could choose to request an iButton in advance of the election, and then take advantage of the ability to vote at any of the available electronic polling places.

5. Acknowledgements

We would like to thank Kathy Richardson and David Jefferson of the Compaq Western Research Lab, Professor B.R. Badrinath of Rutgers University, and Professors Dan Boneh, Mary Baker, and Armondo Fox of Stanford University for their advice and support on this project.

6. References

6.1 Disconnected Filesystems

James J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. In *Proceedings of the thirteenth ACM symposium on operating systems principles, October 13−16, 1991, Pacific Grove, California*. Pages 213−225.

L. B. Mummert, M. R. Ebling and M. Satyanarayanan. Exploiting weak connectivity for mobile file access. *Proceedings of the fifteenth ACM symposium on operating systems principles, December 3−6, 1995, Copper Mountain, Colorado.* Pages 143−155.

Karin Petersen, Mike J. Spreitzer, Douglas B. Terry, Marvin M. Theimer and Alan J. Demers. Flexible update propagation for weakly consistent replication. *Proceedings of the sixteenth ACM symposium on operating systems principles, October 5−8, 1997, Saint−Malo, France.* Pages 288−301.

D. B. Terry, A. J. Demers, K. Petersen, M. J. Spreitzer, M. M. Theimer and B. B. Welch. Session guarantees for weakly consistent replicated data. *Proceedings of the international conference on parallel and distributed information systems (PDIS), September 1994, Austin, Texas.* Pages 140−149.

6.2 Cryptographic Election Protocols

J. Benaloh and D. Tuinstra. Receipt−Free Secret−Ballot Elections. In *Proc. 26th Symposium on Theory of Computing (STOC '94)*, pages 544−553, New York, 1994. A.C.M.

R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi−authority secret ballot elections with linear work. In *Advances in Cryptology − EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72−83, Berlin, 1996. Springer−Verlag.

Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A Secure and Optimally Efficient Multi−Authority Election Scheme. In *Advances in Cryptology − EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103−118, Berlin, 1997. Springer−Verlag.

K. Sako and J. Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptogology − CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 411−424, Berlin, 1994. Springer−Verlag.

K. Sako and J. Kilian. Receipt−free mix−type voting scheme − a practical solution to the implementation of a voting both. In *Advances in Cryptology − EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 393−403, Berlin, 1995. Springer−Verlag.