Practical Distributed Voter-Verifiable Secret Ballot System

Edoardo Biagioni
Dept. of Info. and Comp. Sci.
1680 East-West Road, POST 317
Honolulu, HI 96822

Yingfei Dong
Dept. of Electrical Engineering
2504 Dole St, Holmes Hall 483
Honolulu, HI 96822

Wesley Peterson, Kazuo Sugihara
Dept. of Info. and Comp. Sci.
1680 East-West Road, POST 317
Honolulu, HI 96822

ABSTRACT

In this paper we propose an end-to-end voter-verifiable system, which is more transparent than existing solutions. Its distributed nature permits its use both for supervised voting at a polling place and for online remote voting. We use Chaum's blind signatures to ensure voter anonymity. In particular, voters can verify that their votes are recorded exactly as cast, and all ballots can be made public so that anyone can verify the election results. We further address key security issues such as server corruption in the proposed system. The case is made that this can be the basis of a practical voting system.

Categories and Subject Descriptors

K.4.0 [**Computers and Society**]: General—*Voting Systems*; K.4.1 [**Computers and Society**]: Public Policy Issues—*Anonymity Verifiability*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms

Security, Algorithms

Keywords

voting systems, end-to-end, voter verifiable, secret ballot

1. INTRODUCTION

It has been clearly established that existing direct-recording electronic (DRE) voting machines cannot be trusted [1], [2], [3]. On these systems, votes can be lost or altered, and no effective way is provided for detecting such failures. Therefore, the need to audit or recount has led to the widespread use of paper ballots, either as the primary ballots or as the backup of voting machines. However, paper ballots are vulnerable to forgery, loss, or substitution [4, 12]. The election officials handling the paper ballots at every point must be trusted. A few improved voting systems have become commercially available [5]. Also, several innovative schemes such as PunchScan and ThreeBallot have been devised that permit voters to check whether their votes have been included in the tally correctly [6], [7], [8], [9]. Meanwhile, these systems strongly prevent voters from being able to prove how they voted, to prevent vote selling and coercion. As a result, they are not completely transparent.

In a recent paper, Yasinsac and Bishop [4] make several interesting points. Their first sentence is "At the end of the day, elections are about counting votes." The accuracy and assurance of the vote counting should be the highest priority. They find that paper ballots or receipts, although safer than many DRE voting machines, are lacking in this regard. They state that "Audit trails are essential to voting systems", and also suggest that "Relaxing the 'vote non-provability' principle" in the interest of providing audit information or making the voting process more transparent, may be desirable. We have come to the same conclusions, and in this paper we suggest a voting protocol that is much more transparent than most existing systems and that we believe can be the basis of a practical election system. We call this system Distributed Voter-Verifiable Secret Ballot (DVVSB). DVVSB can be used for both voting at a supervised polling place and remote voting on the Internet, compared with most end-to-end voter verification solutions that are designed for polling stations. It is especially useful to complement increasing popular mail voting [12]. It provides a concrete example of what can be done if the non-provability condition is relaxed.

In the proposed system, we separate voter authentication and vote tallying in a distributed environment, including *an authentication server, a tally server, and voter clients*. We assume that voter registration has been conducted separately and we have a list of eligible voters. This list is provided to an authentication server for voter authentication. We assume that each voter obtains an identifier and authentication keys during the voter registration process. For vote tallying, a tally server collects all cast ballots into a tally database. Each ballot has a ballot identifier randomly assigned by a voter client. Furthermore, for tally verification, the list of eligible voters and the tally databases are made public at the end of an election. In particular, the voter can use the ballot identifier to check whether that ballot is in the tally database, and check the exact contents of the ballot. In principle, anyone can access the entire contents of this tally database and count the ballots themselves. This will give transparency to the vote-counting process and assur-

SAC'09 March 8-12, 2009, Honolulu, Hawaii, U.S.A.

ance that the votes are correctly counted. However, there is no way to find which ballot was cast by which voter. The voters know their own ballot identifiers, but they cannot prove that, because the ballot identifiers are all public and anyone could claim to have cast any ballot.

2. BASIC PROTOCOL

The proposed basic protocol uses Chaum's blind signatures [10] to achieve vote anonymity. Blind signatures have been proposed for use in elections [11]. Our proposed protocol is based on these ideas. Our scheme is similar to [14], but allows the use of multiple, redundant servers, which also prevents a corrupt signing authority from casting ballots for voters who do not cast a vote.

In the basic protocol, each voting operation involves three entities: a *voter client (VC)*, a *ballot authentication server (BAS)*, and a *vote tally server (VTS)*. The BAS maintains an *eligible voter database (EVD)*, which is used for authenticating and recording blinded ballots from voters and tracking which voter has voted. Each entry in the EVD includes a voter identifier, a vote flag indicating whether the voter has voted, and a signed, blinded ballot from the voter if the voter has voted. The VTS maintains a *vote tally database (VTD)* by collecting all cast ballots. Each entry in the VTD includes: (i) a unique ballot number chosen by a VC; (ii) the voter's choices; (iii) a BAS digital signature confirming that this is a valid ballot.

The first step in the protocol is for the election authority to furnish each voter with a voter identifier V_{ID} and secret authentication keys. The keys should be kept secret, because anyone who has them can vote. We have not addressed the problem of doing this securely. We assume that the election authority has a procedure for identifying individual voters and authorizing their voting.

Next, a voter client (VC) on a computer prepares a ballot with the voter's choices and with a randomly-generated ballot identifier B_{ID}. The B_{ID} must be large enough such that the probability is negligible that two voters will by chance choose the same ballot identifier. Then the voter blinds the ballot using a randomly chosen number less than n.

Then the voter client sends the blinded ballot along with V_{ID} to the BAS. The server checks whether the V_{ID} is in the EVD database and whether the voter has voted. If the voter has not voted, the server signs the ballot, stores this signature in the EVD database with V_{ID}, and also sends the signature back to the voter client. If the database indicates that the voter has already voted, the server sends the voter client the previously sent blinded, signed ballot stored in the database, in case the voter did not receive it on the last transmission.

Upon receiving the signed, blinded ballot, the voter client unblinds it; the result is exactly the signature of the original ballot. Then the voter client verifies the signature. If it is correct, the voter sends the signature along with the unblinded ballot to the tally server VTS.

After the polls have closed, the entire contents of both the eligible voter database and the vote tally database are made public. First, since the entire contents of the tally database are public, any voter can check whether their votes are included in the database correctly, using their ballot identifiers. Secondly, anyone (who has the computer expertise) can count all the votes in the tally database and know the results of the election. Furthermore, since the eligible voter database is also public, anyone can see who voted. However, there is no reliable information to tell which ballot corresponds to which voter from the contents of the two databases. We will discuss in later sections how to deal with compromised authentication and tally servers by splitting the authentication and tally responsibilities among multiple servers.

Table 1: Notation.

A	original ballot
B	blinded ballot
C	signed, blinded ballot
D	signed ballot
n	RSA modulus
e	public key
d	private key
k	blinding factor

Here are the mathematical details of using blind signatures for voter authentication and anonymity. Blind signatures use RSA. In this section, assume that n is an RSA modulus and e and d are the public and secret keys, respectively. Then for any X, if $C = X^e \bmod n$, then $X = C^d \bmod n$, and similarly, if $C = X^d \bmod n$, then $X = C^e \bmod n$. The notation is summarized in Table 1. We use A to denote an original ballot prepared by a voter client. The voter client selects a random number, kept secret, $k < n$, that has an inverse k^{-1} modulo n, and blinds the ballot by multiplying A by k^e modulo n. The result is the unsigned, blinded ballot $B = Ak^e \bmod n$. The ballot authentication server then signs B. We denote the signed, blinded ballot by C. Then $C = B^d = (Ak^e)^d = A^d k^{ed} = A^d k \bmod n$ (since in RSA, $ed = 1 \bmod n$). To unblind this, the voter client multiplies this by $k^{-1} \bmod n$, and the result is $D = A^d \bmod n$, which is exactly what would have resulted if the authentication server had directly signed A. However, the authentication server has not seen A and has no knowledge of the fact that it has signed A. It is this value $D = A^d \bmod n$ that the voter sends to the tallying server. The tallying server then verifies the signature from the authentication server by calculating $A = D^e \bmod n$, and can then tally the voter's choices from the content of A.

Voter Privacy. Now let us consider whether making public the complete contents of both the eligible voter database EVD and the vote tally database VTD reveals any information about which voter cast which ballot. To that end, let us choose any entry $\bar{C}$ in the EVD database and choose any entry $\bar{D}$ in the VTD database. Let us ask whether the voter corresponding to $\bar{C}$ might have cast the ballot $\bar{D}$. Define $\bar{k} = \bar{C}(\bar{D}^{-1})$. (If $\bar{C}$ and $\bar{D}$ belong to the same voter, then $\bar{k}$ is the randomly chosen value that the voter used to blind their ballot.) Now consider what would have happened if a voter had prepared a ballot $\bar{A} = \bar{D}^e$ and blinded it using $\bar{k}$ giving $\bar{B} = \bar{A}\bar{k}^e$. It would be sent to the authentication server and signed, giving $\bar{A}^d\bar{k}$ which is equal to $\bar{C}$, because $\bar{C} = \bar{k}\bar{D}$ by the definition of $\bar{k}$ and $\bar{D} = \bar{A}^d$. Now if this is sent back to the voter client and unblinded by multiplying by $\bar{k}^{-1}$, the result is $\bar{D}$. Thus a voter starting with ballot $\bar{A}$ and using the "random" blinding factor $\bar{k}$ would have caused the entries $\bar{C}$ and $\bar{D}$ to appear in the databases. Therefore, there is no evidence that $\bar{C}$ and $\bar{D}$ did belong to the same voter. Whether $\bar{C}$ and $\bar{D}$ belong to the same voter or not, it appears possible that they did. Thus knowing both entries tells you nothing about whether they correspond to

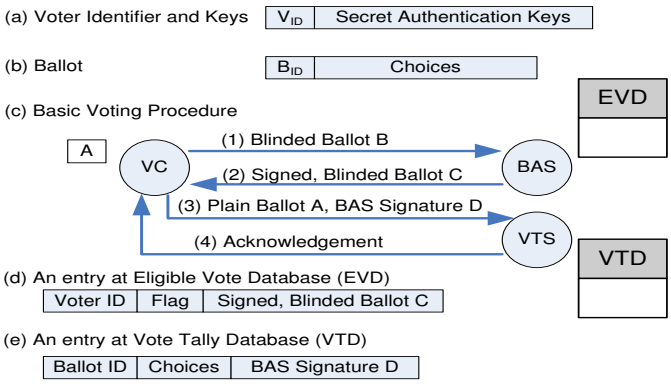

Figure 1: Basic Data Structures and the Voting Procedure. (a) Voter identifier format. (b) Ballot format. (c) Basic voting procedure. (d) Entry format in the eligible voter database. The flag is initialized as 0; it is set to 1 when a voter has voted. (e) Entry Format in the vote tally database.

the same voter. There is no way for an observer to tell from looking at the two databases whether $\bar{C}$ and $\bar{D}$ correspond to a voter or not. Moreover, there is no way for an observer to tell whether any pair of entries like $\bar{C}$ and $\bar{D}$ correspond to a voter.

(If $\bar{D}$ has no inverse modulo n, then $\bar{k} = \bar{C}(\bar{D}^{-1})$ doesn't work. However, it is still possible to find a $\bar{k}$ such that $\bar{D} = \bar{k}\bar{C}$ using the Chinese Remainder Theorem, and the rest of the argument follows.)

Open Source Implementation. The proposed algorithms and protocols are simple and clear. The software, algorithms, and the file formats should all be open-source. This will increase the transparency of the system and will result in no loss in security. In addition, software for checking and counting ballots could be produced by anyone, for example by the political parties or the League of Women Voters, and that could provide a check on the correctness of the "official" software. We will build a prototype system and make it public.

3. ENHANCING SECURITY WITH REDUNDANT SERVERS

The single BAS and the single VTS in the basic protocol are two potential single-point failures. Now we address these issues by sharing the authentication and tally responsibilities among multiple BAS's and VTS's.

3.1 Redundant Authentication Servers

We use redundant BAS servers to address the potential failure of the authentication server. For the sake of a simple discussion, let us assume that there are four authentication servers. Each server has its own set of RSA keys. The voter has an identifier, which is supplied to all authentication servers and may be public after the election for auditing. When the voter receives permission to vote, that includes a key for each server, four keys, which must be kept secret. The voter client sends blinded copies of a ballot to any three of the servers, using a different blinding factor k for each. After receiving the three signed, blinded ballots, the voter client unblinds them and checks them and if they are correct, sends the three signatures to the tally server, which checks all three signatures and then puts the ballot in the database, with the signatures. (The tally server now requires three signatures from three different authentication servers for each ballot it accepts.)

At the end of the election, the data from all four servers is merged to make a list of identifiers of all voters who requested that blinded ballots be signed.

This modification protects against three potential problems.

1. *Single BAS failure.* If one BAS server fails, or if one of the signatures that the voter's client computer received fails to check, the client can request a signature from the fourth authentication server to replace the faulty signature. Thus this protects against the failure of any single server.

2. *Vote insertion by a compromised BAS.* Suppose one BAS server is compromised. There is not enough information in any one server to determine who has voted and who has not, and thus to make it possible to submit invalid votes for persons who failed to vote legitimately. Besides that, simply knowing the identifiers for voters who have not voted would not be enough information to permit illegal votes to be entered for them. One would have to know the keys for each such voter for at least two other servers besides the compromised one.

3. *Collaboration between a compromised server and a voter.* Suppose that a voter had to submit only two signatures to the vote tallying server VTS. Here is an interesting attack: Suppose that one authentication server is completely compromised, so it can sign blinded ballots without checking whether or not the submitter has already voted. Then suppose a voter collaborates with this server. The voter can vote three times, by getting each time a signature from the compromised server and one of other servers, and thus getting sets of two signatures three times. With three signatures required from four servers, this is impossible because the voter attempting to cheat could get signatures from the compromised server and two other servers the first time. The next time the voter could get a signature from the compromised server, but then only from the one remaining server.

More generally, it is possible to protect against m or fewer compromised or failed authentication servers by providing $3m+1$ authentication servers and requiring $2m+1$ signatures for a ballot to be valid at a tally server.

3.2 Redundant Tally Servers

Obviously having a single tally server is another weak point in the basic protocol, but a very easy one to fix. Simply having multiple tally servers protects against the failure of any single server. However, there are a lot of possibilities for protecting against failures here.

1. There can be multiple tally servers, and the voter client program can require verification from more than one tally server. Even if one tally server is completely compromised, it cannot spoil the election by inserting invalid data into the tally database, because all of the

ballots can be verified by the signature values and only ballots with correct signatures should be kept.

2. A printed receipt can be made with the votes and the ballot identifier and the signatures all printed but with no identification of the voter. A query program can be made as part of the tally server to verify votes. If anyone enters the ballot identifier, the vote will be shown, thereby confirming that it is in the database. Now there is the problem of selling votes. If the system allows queries of ballots and allows anyone to print any ballot (with ballot identifier and signatures) then no one can prove that a printed receipt is that voter's own vote, so redundant receipts would make any receipts have little value to sell.

3. There is an excellent idea suggested by Simha and Vora [6]. The suggestion is that a voter have the option after voting of sending the same ballot as was sent to the tally server, to a third party, possibly with more than one choice. The third parties could check at the end of the election whether that ballot is in the tally database, and if not, submit it.

4. At the end of the election, the authentication servers stop signing ballots, and publish the record of which voters voted. At this time, the tallying servers must publish the list of votes received. If any voter (or private tallying server) is unable to locate the vote within the database, the vote can be resubmitted and should be accepted if it is properly signed. Any ballot that is properly signed must be valid, and the tallying server can easily check whether the ballot is already in the database.

   This second phase may be ended at a specific time, or as soon as the number of ballots unaccounted for is insufficient to alter the outcome of a given race. The number of ballots not yet submitted can be determined by comparing, to the number of ballots in the tallying servers, the set of voters for which each authentication server has signed ballots. This information is only available after the BAS have published the list of voters for whom they signed ballots.

5. Since both the vote tally and the record of whose votes were signed are published, anyone can count the votes and anyone can check whether the number of votes in the tally database is consistent with the number of voters who have requested signatures on blinded ballots.

Both a ballot authentication server and a vote tallying server can be implemented to take advantage of multicore processors. In particular, illegal packets can be detected and discarded in parallel, and the processing of valid packets can also proceed in parallel. The only sensitive point is in the recording of valid packets. This requires the authentication server to briefly lock only the record corresponding to the voter whose vote is being recorded. The tallying server can have the same requirement, or simply save all valid votes and allow a post-processing phase to detect duplicates. In short, processing of packets on the servers can be expected to speed up nearly linearly with the number of processors.

4. SECURITY AND PRACTICALITY

We feel that this proposed system can be the basis of a practical, usable voting system with some advantages over other existing and proposed systems. We have implemented a skeleton version of the cryptographic and network protocols using 2048-bit RSA for the principal algorithm. Our results on a 2.8GHz Intel Core 2 Duo indicate that a server on a common PC can handle at least a couple hundred votes per second. The rest of this section contains a discussion of several security issues and practical ways of dealing with them.

The first step in the process is for the election authority to prepare a list of eligible voters and to provide an identifier and four keys (for the four authentication servers) and then to send to each authentication server a list of all voter identifiers with the key for the server for each identifier. Obviously, this must be done confidentially and securely. Steps can be taken to secure the computers involved. For example, they can be isolated from the Internet. Obviously this must be done by trusted personnel.

The second step is to verify the identification of voters and give them their identifiers and keys, e.g, on a flash device at a polling place or on a CD for remote voting. Identification can be done in the way it is presently done. For example, at a polling place an election worker can examine the voter's picture identification.

Next the voter uses a client computer to do the voting. The computer must be secure. The program should be carefully designed and open source. It might be directly booted from a secure CD. A compromised client might make the voter's choices not secret. However, if the ballot that it prepared is not correct, the fact will be detected. The ballot is valid if and only if it consists of the voter's choices, a ballot identifier, and valid signatures. The ballot's validity can be easily verified by another independent program. Also voters can check whether their ballots are correctly contained in the VTD online.

The voter client prepares a ballot. Then using a different random blinding factor each time when a blinding factor is needed, for each of three authentication servers, it blinds the ballot and sends the blinded ballot, the voter identifier and the key to the server. If this transmission is over a network, it would be subject to a man-in-the-middle attack. The blinded ballot is for all practical purposes encrypted, but the voter identifier and the key are not. Some kind of encryption or authentication is required. The voter's key could be used for that purpose. For example, in our current implementation, the key is used to compute a SHA-512 HMAC over the ballot and ID, which allows the BAS to quickly detect packets without a valid key, or packets sent with a key that has already been used. The transmission of the blinded, signed ballot back from the server to the voter client would not need to be encrypted–being blinded, it is for all practical purposes encrypted, and being signed, it would not be subject to alteration. If the voter client can verify the signature, the signature must be correct.

With four authentication servers, the voter is protected against the failure or compromise of any one authentication server. If one of the requested signatures fails, the voter client can ask for a retransmission. If that fails, the voter client can ask the fourth authentication server for a signature–three signatures are required for a vote to be accepted by the tally server.

This is a critical point in the process. Once the authentication server has signed a blinded ballot and sent it, that ballot cannot be canceled, because the authentication server has no way to know that the signature has not been unblinded and used with a ballot sent to the tally server.

The voter client can verify the three signatures and compare with the original ballot, and so the voter client can know that it has constructed a valid ballot. This can then be sent to at least two tally servers, and the voter client should not consider the vote cast until it receives signed acknowledgments from the tally servers. Having the tally servers sign the acknowledgments would assure that the acknowledgments came from the real tally servers, and not, for example, from a man-in-the-middle attacker. The voter client can print that ballot–the ballot identifier, the voter's choices, and the three signatures–and this printed page could be submitted later to be counted if the ballot does not appear in the database of ballots. In addition the voter client can offer the voter the option of sending the ballot to one or more third parties who can collect these ballots and later check whether they are in the database of ballots and if not submit them to be counted.

With the scheme described by Cranor and Cytron [14], the single Validator (equivalent to our BAS) knows, towards the end of the election, how many voters are entitled to cast votes but have not yet done so. A corrupt Validator could sign and submit votes for such voters. If a voter who doesn't vote is unlikely to check whether their vote was cast, this may not be detected by individual voters. With our scheme, a single corrupt BAS can sign votes, but cannot get two other BAS to also sign its votes, as would be needed for a valid ballot.

If an attacker observes the network traffic to and from a voter client, they might be able to determine how that voter voted by associating the traffic to the authentication server with the traffic to the tally server. For this reason the communication from the voter client to the tally server should be encrypted using the tally server's public key. If the voting authority or the voter prefer to protect against an attack by a compromised tally server *and* someone intercepting communication, the ballot can be anonymized by being sent encrypted to one or more third parties that the voter trusts, such as the League of Women Voters or a political party. Such third parties could decrypt the anonymized ballot and forward it to the tally server.

The tally server, upon receiving a ballot, should verify the three signatures and if they verify, check the ballot identifier to see whether that ballot identifier is already in the database. Then it should place the ballot in the database. It should, basically, accept properly signed ballots from anywhere without asking whose they are or where they come from. It would be practically impossible to construct a set of ballot signatures without knowing three of the authentication servers' secret keys. (If the authentication servers' secret keys are compromised, the election is compromised.) After the polls close, there should be a period of time allowed for checking by individuals or third parties whether valid ballots they have are in the database, and if not the tally servers should accept them. The entire database should be made publicly available immediately after the polls close. This will make possible checking whether ballots are in the database. It will also make it possible for anyone to count the votes and determine the outcome of the election.

After the polls close, the list of voters who voted can also be released. (The entire databases from all four authentication servers can be made public.) From the tally server database, it is possible to count how many signatures have come from each authentication server. This should not exceed the number of signatures each authentication server has recorded in the EVD. (Some signatures may not have been used for some reason or other, such as a voter's failing to complete the voting process.)

Denial of service attacks on the Internet must be considered. For voting at a designated polling place, at least a subset of the authentication servers and one tally server could be located at the polling place on a local area network that does not allow unsolicited traffic from the Internet, and therefore is not subject to a DOS attack. Even when used on the open Internet, the system is designed to be resistant to all but the most intense DOS attacks, since: (1) The voting client only needs to exchange a relatively small UDP packet with each server, and such a packet is more likely to get through than the more common TCP traffic. (2) Since the algorithm works even in the presence of failing or corrupted servers, it can transparently accommodate the loss of some of the packets. (3) Packets are idempotent, and may be retransmitted as long as needed until a corresponding acknowledgment is received.

With all the data redundantly stored and publicly available as in this system, dependence on a paper trail is very greatly decreased.

5. CONCLUSION

In the past 12 years or so, there have been some very novel and very good proposals for secure election systems. To the best of our knowledge, none of them has been used in a major election. In the election this year, it appears that most votes will be counted using paper ballots scanned into a computer or DRE's. Some paper ballots may still be counted by hand and some lever-type voting machines may still be in use. Why hasn't the latest technology been put to use?

We believe that by shifting the priorities on what must be most secure, we have been able to define a voting protocol DVVSB that has a greater transparency than currently used and proposed systems, and yet has strong cryptographic security. It can be used both with voting at a polling place and over the Internet. This proposed system provides a concrete example for insight into Yasinsac and Bishop's suggestion that vote non-provability might be relaxed in the interest of gaining greater transparency in vote counting. We are currently implementing the complete prototype system. We are planning to deploy it in a real election to gain the first-hand experiences of the security and performance challenges, such as a school election or Honolulu Neighborhood Board election.

6. REFERENCES

[1] Science Applications International Corporation (2003). *Risk Assessment Report: Diebold Accuvote-TS Voting System and Processes*, September 2, 2003.

[2] D. Bowen, Top-to-bottom Review. Available at http://www.sos.ca.gov/elections/elections_vsr.htm

[3] K. Yee, "Building Reliable Voting Machine Software," Ph. D. Dissertation, UC Berkeley, (2007).

[4] A. Yasinsac and M. Bishop, *Of Paper Trails and Voter Receipts*, Proceedings of the 41st Hawaii International Conference on System Sciences (Jan. 2008).

[5] G. Beroggi, "Secure and Easy Internet Voting,", Computer, Vol. 41, No. 2 (Feb. 2008), pp. 52-56.

[6] R. Simha and P. Vora, "Vote Verification using CAPCHA-like primitives," IAVoSS Workshop On Trustworthy Elections (WOTE 2007) University of Ottawa, Ottawa, CANADA June 20-21, 2007.

[7] B. Adida and R. Rivest, "Scratch & Vote: a self-contained paper-based cryptographic voting," Proc. of the 5th ACM Workshop on Privacy in the Electronic Society pp 29-40. ACM Press (2006).

[8] David Chaum, Peter Ryan, and Steve A. Schneider, "A practical voter-verifiable Election Scheme," Technical Report CS-TR-880, School of Computer Science, University of Newcastle upon Tyne, UK. (2004).

[9] R. Rivest and W. Smith, "Three voting protocols: ThreeBallot, VAV, and Twin," Proceedings of the USENEX/ACCURATE Electronic Voting Technology Workshop (ETV2007). USENIX Press.

[10] D. Chaum, "Blind signatures for untraceable payments," Advances in Cryptology, Proceedings of Crypto 82. Plenum Press (1982). pp. 199-203.

[11] B. Schneier, Applied Cryptography, 2nd Ed. Wiley (1996). Using blind signatures for elections. pp. 126-127.

[12] A. Yasinsac and M. Bishop, "The Dynamics of Counting and Recounting Votes," IEEE Security and Privacy, Vol.6, No.3, 2008, pp. 22-29.

[13] D. Chaum, A. Essex, R. Carback, J. Clark, S. Popoveniuc, A. Sherman, and P. Vora, "Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting," IEEE Security and Privacy, Vol.6, No.3, 2008, pp. 40-46.

[14] L. F. Cranor and R. K. Cytron. "Sensus: A Security-Conscious Electronic Polling System for the Internet." Proc. of the Hawai'i International Conference on System Sciences, January 7-10, 1997.