Undervote and Pattern Voting: Vulnerability and a mitigation technique

Stefan Popoveniuc and Jonathan Stanton*

Abstract

Highly secure and auditable voting systems have been developed recently that open up the possibility of not only voting systems that claim to be secure, but ones that can actually be verified as secure and audited to the satisfaction of third party observers. Yet, these systems are still in development and do not yet handle all of the cases that arise in real-world elections. This extended abstract presents one specific vulnerability or limitation that arose in several of these voting system: Undervote and pattern vote tracking; This issue arose not so much because of flaws in the systems themselves, but rather, they are cases in which the strong protections provided by these systems did not take into account behaviours such as not voting on every contest. We present these issues in the context of PunchScan, a state of the art voting system.

We present a technique for solving the issue using contest partitioning and analyze their impact on performance.

1 Introduction

The development of cryptographic and information theoretic based voting systems has been a major development in making possible not only more secure voting systems, but also rigorously verifiable and auditable systems that can be shown to maintain the integrity of the vote count as well as good individual privacy to the voter even in the face of glitches, malicious attacks and human error. These relatively new voting systems (although based on techniques that go back two decades[Cha81, Ben87]) are still maturing and need enhancement in order to deal with some of the real-world voting constraints of public elections.

The PunchScan voting system was initially described by David Chaum[Cha05] in a presentation he made at the Workshop on Frontiers in Electronic Election 2005 and attracted a lot of attention because of its potential to provide provable security guarantees without the complex ballot formats required by previous systems and in such a way that all citizens can understand both how the voting system worked and why it is secure. An initial version of the PunchScan system was developed by Chaum and a group of students at several universities in the US and Canada.

This paper introduces one specific vulnerability or limitation in the PunchScan voting system and describes techniques that correct the weakness. Although not all other voting systems, besides PunchScan, have this specific vulnerability, many do. The technique presented here could be adapted to enhance other systems as well.

The vulnerability is a weakness in the voter's privacy. It was determined to be possible to trace a ballot back to the citizen who voted it by examining the undervotes on the ballot.

*Authors (poste@gwu.edu, jstanton@gwu.edu) are with the George Washington University, Department of Computer Science, Washington DC 20052. Stefan is also a member of the PunchScan team

It should be noted that this limitation is not the result of a flaw in the main components of PunchScan, rather, it is a case in which the strong protections provided by PunchScan did not extend to some real-world scenarios or did not take into account methods of voting (such as not voting) that are usually permitted in elections.

The solution to this vulnerability is to divide the ballot into smaller ballots (called groups), and to have one separate set of transformation and result for each group. At one extreme would be to have each contest in its separate group, while at the other extreme would be to have all the contests in one group (our first approach).

2 Review of PunchScan

A detailed description of PunchScan can be found in [PH06], with an overview of how different entities interact with the system being provided in [FCS06].

3 Related Work

Since surveys and overviews of voting systems [Pro01, Rja03] are easily available, we do not provide an overview here. Voting system designs have appeared in several other papers, such as [Cha04, SK95, SK94, Ben87, CGS97, FOO93, Oka97].

Recently several secure voting systems have been proposed: CVV [Vor05, HPS+05], PunchScan (PS) [Cha05, PH06], VoteHere (VH) [VI03, Nef01], Pret-a-Voter (PV) [CRS05], Scratch&Vote (SV) [AR06], and ThreeBallot (3B) [Riv06] all having end to end integrity and verifiability.

All the voting systems that present individual clear text ballots are susceptible to fingerprinting attacks. In particular CVV and Pret-a-Voter have fully decrypted ballots at the exit point of the mix-net. Scratch&Vote and ThreeBallot do not present clear text ballots, rather they use homomorphic encryption to compute the tally. One of the known weaknesses of ThreeBallot is it's susceptibility to fingerprinting each strip of the ballot. Our technique of dividing the ballot into smaller groups would not alleviate the problem for ThreeBallot because the fingerprinting can be done within a particular question.

A more detailed discussion of how PunchScan works can be found in [PH06]. How to evaluate the impact on integrity of a voting system is described in [HV06].

4 Undervote Ballot Tracing

The ability to trace a ballot to the voter who cast it by examining information on the ballot is a well known risk in traditional voting systems. Since all of these techniques require the cooperation of the voter themselves, they are mainly a concern because of the possibility of vote-buying and voter-coercion. For example, on a paper, hand-counted ballots if a voter can write-in an arbitrary candidate name for a certain office, they could write in their own name for that office, or a pre-determined keyword if their vote was being purchased. Then during the count of ballots, anyone who can view the ballots (officials, observers, etc) can see the voters name or the keyword and verify that they voted correctly (as the voter was instructed to vote). Using write-in candidates to uniquely identify ballots is a problem in any voting system which allows the write-in marks to be coupled with the rest of the ballot. However, even if write-in votes are prohibited or completely separated from the main ballot, forms of ballot tracing are still possible and are the main focus of this paper. Since write-in votes can easily be separated from the rest of a ballot (whether on paper or

id	P_1	P_2	P_3
1			1 0 2 0
2			1 1 0 2
3			0 1 0 2
4			1 1 0 2
5			0 2 0 0
6			2 1 0 1

D_1	D_2	D_3	D_4	D_5

R_{id}	R_1
1	1 2 1 0
2	1 2 1 0
3	2 1 0 0
4	1 1 2 1
5	0 1 1 0
6	1 2 1 0

Figure 1: Example of fingerprinting a ballot using patterns of votes.

in electronic form) and processed separately (in the same way that absentee ballots are processed) or write-in votes can be restricted to only pre-approved *write-in candidates* names this specific form of vote-buying has possible solutions independent of any particular voting system.

In a slightly more complicated, but less obvious, method, the voter may mark a specific pattern of votes, say for an office of relatively less importance (for example "Town Dog Catcher") but which has several candidates running, while voting for the instructed candidate for the major contests. Again, someone who can observe the votes, can recognize the pattern of votes made in the "Dog Catcher" contest and know this is a purchased or coerced vote and verify that the major contests are voted correctly. Clearly this technique provides less information/identity leakage as the number of bits of information that may be encoded in a a voting pattern for a typical contest may be limited to distinguishing between 5-10 different patterns.

For example in Figure 1 we show an example of this technique in PunchScan. Here, Alice runs in the first contest (and is candidate number one). She has coerced Victor, Valentino and Valerie to vote for her for contest one, and instructed them to vote for the third, second and first candidates in contests two, three and four respectively. After the election is complete, Alice can look in the R table and see that there are three ballots that have those exact vote patterns (rows one, two and six in the R table). She also sees that there is a vote for her in each of those three ballots. Thus she can verify that the coerced voters cooperated. However, if she only finds two ballots in the R table with the correct pattern, she will not know which of Victor, Valentine or Valerie did not comply.

A second vulnerability exists because of the way in which undervotes are handled. In almost all elections, voters are allowed to "not vote" for any particular office – and in many of those cases they do not have to mark something on the ballot to indicate they are not voting for that office, they can just leave it blank. This is called an *undervote* as the voter did not vote as many times (or for as many offices) as they were permitted. Because of how PunchScan records votes on the ballot, any undervote is immediately recognizable on both the top and bottom pages — it is an office with *No marks* on any of the circles or letters. Thus, unlike any other selection a voter can make, with an undervote someone who collects the ballots for scanning or simply observes the scanned images (which are public) can see on which offices a particular person undervoted.

For example, in Figure 2 we see Alice again trying to buy the election. She is running in the first contest as candidate number one. She has coerced three voters to vote for her and she told Victor to abstain in contests 2 and 3, Valentino to abstain in contest 2 and 4 and Valerie to abstain in contests 3 and 4. After the election, Alice can look in the R table and see that there are three ballots that have those exact positions undervoted. She also sees that there is a vote for her in each of those three ballots. This attack is even more powerful because Alice can link these ballots from the R table back to the cast ballot in the P table, which retains it's serial number. Thus if a coerced voter does not mark the required undervote pattern, their serial number can be determined and then Alice can

id	P_1	P_2	P_3
1			1 0 2 0
2			1 1 0 2
3			0 -1 -1 2
4			1 1 0 2
5			0 2 -1 -1
6			2 -1 0 -1

D_1	D_2	D_3	D_4	D_5

R_{id}	R_1
1	1 -1 -1 1
2	1 -1 0 -1
3	2 1 0 0
4	1 1 2 1
5	0 1 1 0
6	1 2 -1 -1

Figure 2: Example of fingerprinting a ballot using undervotes

ask each coerced voter to show their receipt, matching serial numbers and the requested undervote pattern. This allows Alice to verify who did not vote 'correctly' when multiple voters were assigned the same pattern. In practice the serial number is often linkable to the voter's identity, for example because the poll-workers see both, or the same computer systems process both registrations and ballot scanning or generation. Whether or not these two are linked does not affect the existence of the attacks.

Thus, the best way for a coerced voter to avoid voting as instructed, is for them to vote the undervote pattern, but actually vote for someone else in the first contest. As long as there is at least one other person voting the same undervote pattern, they cannot be distinguished. How useful this is depends on how many other voters are assigned the same pattern.

To measure how dangerous this undervote attack actually is, one must consider the ability of the attacker to accurately trace individuals to their coerced votes. If only a few undervote patterns are possible, the coercer has a limited choice of either only coercing a few votes, which won't really affect the election, or coercing a lot of votes but having no way to verify or enforce each one. So to analyze the power the attacker has, we need to minimize the number of undervote patterns possible.

As a result, on a ballot with only a few offices, for example many European elections with only members of parliament or maybe a local mayor on the ballot, this risk is not substantial as each office only allows one bit of information to be leaked (did you undervote for office X or not), giving a total of only a undervote patterns. However, with ballots containing many contests and issues, such as in the United States, the amount of information that can be leaked becomes meaningful.

We define a measure U which represents the number of possible ways to fingerprint a ballot using undervotes. On a ballot with q contests, and with m_i possible candidates' choices for each contest:

$$U = \prod_{i=1}^{i=q} 2^{m_i}$$

If only one candidate can be chosen for each contest, $U = 2^q$, thus if $q = 12$ and there are 4092 ballots in total, all of them can be uniquely fingerprinted and traced. This would ruin privacy for all voters who undervote in a known pattern. In practice, fewer fingerprints will actually be possible for a given number of contests, as in vote-buying or coercion some of the contests need to actually be voted for. So some of the contests can not be undervoted to form a fingerprint. But one can easily imagine that 2 or 3 top-of-the ballot contests could be worth vote buying even if it forced the rest of the contests to be voted in a specified pattern of undervotes. This would only decrease the number of fingerprints by a few times to 512 or 1024, still enough to be considered a serious attack.

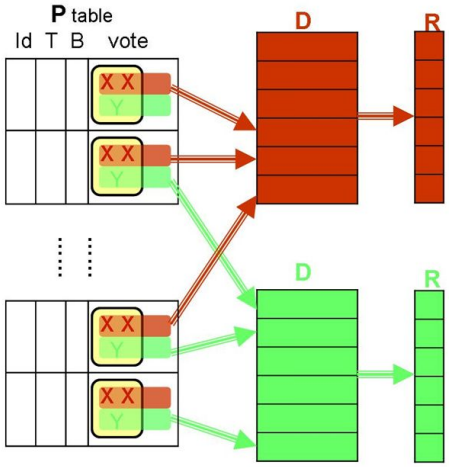

Figure 3: Partitioning a ballot into red and green questions which are processed separately

4.1 Unlinking election contests to prevent undervote fingerprinting

We describe here a simple way of avoiding fingerprinting ballots using patterns of undervotes. Since the pattern of undervotes only provides a useful fingerprint if it is composed of the undervote records from a number of contests, the key idea is to separate the processing of contests into different operations that will unlink the contest votes from the same original ballot into separate votes for only a particular contest. This unlinking will mean that the set of votes for contest 1 (say President) can not be associated with the same voters votes for other contests such as Senator.

This type of unlinking is quite unlike traditional paper ballots, or optical scan systems, where the physical ballot is kept as a single, unified piece of paper during the vote counting, recount, and post-election processing. So how you voted on contest 1 is clearly linked to how you voted on contest 2. Now, other voting technologies such as DRE or Mechanical Lever machines do not have this property of linkability of contests as they do not record a 'ballot' in the traditional sense, but only store the tallies of votes for each office. In their case, while avoiding the contest linking they also avoid any possibility of meaningful recounts or ballot auditing, while PunchScan can unlink the contests while preserving the strong auditability of cryptographic and paper ballot systems.

To unlink contests in PunchScan, the use of the D table must be modified, as undervotes are visible on both the input and output sides of the D table (from the P table to the R table) and thus allow linking the scanned images to final clear-text ballots.

We propose to partition the ballot into sections, each section having it's own set of D tables and its own results table.

For example, in Figure 3 we show a simple example of a ballot with two sections, one labeled in red and one labeled in green. The full submitted ballot is stored in the P table, however, each portion of the ballot that corresponds to a particular partition is linked to a different D table, so the red responses can no longer be linked to the green answers when they reach their respective R tables as the mixing that occurs in the links from P to D and D to R is different for each partition. It is just as if we were having separate elections (one with a red ballot, another one with a green ballot), while allowing the voter to check his receipt from a single place (a single P table).

At the extreme, each section contains only one contest. This would impact performance, since each row in each D table needs to have a commitment, and the number of D tables get multiplied

by the number of sections.

For a single section the number of ways to fingerprint a ballot is:

$$U = \prod_{i=1}^{i=s} 2^{m_i}$$

where s is the number of questions in the section. Analyzing the maximum number of candidates that can be selected for each contest, the ballot can be partitioned in sections such that the value of U does not get too high for any section.

The use of separate D tables for each section results in the original ballot, with all of the response marks on it, being split into P separate groups of contests and each contest group being decrypted into a clear-text vote through it's own D table. Since the clear-text votes for each contest are now stored separately and can not be linked to the clear-text votes for other contests that were made by the same voter.

Since each separate D table stores commitments for the linkage between the scanned ballots and a clear-text result for one (or a few) contest, if privacy is lost on a particular D table, the only information revealed is how a voter voted on that specific contest. No information about other contest votes is released since the linkage between those contest results and the original ballots are protected by a separate D table with it's own commitments and permuted relationships. The integrity of the election is also not weakened, as this technique only separates contests within the ballot, so the same audit procedure is employed (but now for more tables) and the same risk of exposure exists for malicious election authorities.

Contests can not be unlinked arbitrarily, as sometimes there are cases where contests are naturally coupled. For example, in an impeachment vote, it may be required that in order to vote on who should replace the current office holder, you must also vote to impeach the current office holder (whether such a rule is good public policy or not is a question we do not discuss). So with the two contests:

1. Do you want to impeach the mayor?

2. Who do you want as new mayor if the old mayor is impeached?

a voter can answer the second question only if he answered "Yes" to the first one. This type of coupled contests can not be enforced if the contests are processed separately. Consider, when the ballot is scanned, just before casting it, such couplings cannot cannot be checked, since there is no way to know what the voters choice was for question 1 (Do you want to impeach the mayor?). The only place where the choice will be visible is the results table. If the results tables for the two contests are unlinked, then it will be impossible to correlate the votes from the same ballot that appear in different results tables, so enforcing the restriction on only voting in question 2 if you voted *yes* for question 1 is impossible. Thus coupled contests must appear in the same results table, thus must belong to the same partition of contests.

Information in coupled contests may be revealed in other ways through undervotes. For example, just the existence of a mark for the second contest indicates that the voter probably chose "Yes" to question 1. Correspondingly, if there is no mark for the second contest, the most likely answer to the first is "No". These tight relationships between contests can not be fixed by the technical changes we propose here, but can most easily be fixed in the design of the ballot by having an explicit no vote for the second contest.

A special case of coupled contests is the ability to cast straight-party ballots. The same problems exist in this case, thus all the contests related to straight-party ballots must appear in the same

partition. Otherwise, double voting may occur. For example, a voter marked his ballots as straight-party, and then votes with the candidates of the same party for all the contests. If each contest belongs to a different partition, then a candidate from that party may get a vote from the straight party question and another from the individual contest, from a single ballot. In this case the problem can be addressed at the scanning station, as if the voter made a mark in the straight-party contest section, the scanner can verify they made *NO* marks in any of the individual contests covered by the party vote. This works as long as the straight-party vote contest is only marked if a voter chooses that option, and is left blank if they plan to vote individually. This is a case where undervoting makes PunchScan's ballot verification easier, not harder.

A slightly different way of fingerprinting the ballot is to vote for a certain pattern of candidates in contests that may be considered unimportant. These patterns may uniquely identify a ballot and thus also reveal the votes for all the contests. The current proposal also solves this problem by unlinking the contests on a single ballot. Now, just like the pattern of undervotes being broken, the pattern of votes in unrelated contests is also uncoupled.

This solution does not affect the integrity or the overall privacy of the election. Let I be the probability that the election authority cheats on at least one vote and is not caught and let P be the size of the crowd a voter can hide in (the size of the privacy set). Then, the grouping of contests does not modify the values of I or P. In PunchScan, as presented in [PH06], I remains $\left(\frac{1}{2}\right)^d$, and P remains equal to B. An advantage of having partitions is that each group of contests can have a different number of D instance tables. The integrity in some contests may be considered more important then in others, thus higher number of D tables can in used.

4.2 Application to other Voting Systems

Contest partitioning can be applied to solve the problem of undervote tracing in other voting systems based on mix-nets, where encrypted and plain-text ballots are published and the decryption process leaves the undervotes unchanged.

5 Consequences of Contest Partitioning

Contest partitioning will affect two aspects of the voting process. First, it will change the time required to process the ballots as additional tables (in PunchScan) or additional mixes (for other mix based systems) will be required. Second, it will change the type of clear-text ballot results that are displayed and available for later study and research after the election.

From a performance point of view, partitioning introduce a cost to generate and maintain the additional tables. This adds a factor of s to how many D tables must be generated and committed to. However, these costs are not the most substantial computational cost in PunchScan which is the generation of mark permutations for ballots, so the overall cost in time does not increase substantially.

In recent elections, the post-election analysis of the voting patterns between different races in the same precincts has been used to uncover potential evidence of irregularities in voting equipment or procedures[KHR06, oVSC06] because expected patterns of voting, such as similar levels of undervotes for different precincts on the same contest, did not appear. Such analysis would be more difficult or impossible if contests were partitioned in the way we propose above, as only totals from specific races would be available, but not correlations between how voters voted on certain races, such as State Governor and US Senator which are both state-wide races.

We present here an overview of two additional versions of the PunchScan partitioning technique

that can actually allow statistical vote pattern analysis while still protecting against undervote or pattern vote attacks. These modifications trade off the ability to do post-election analysis with either a reduction in the privacy set of the voters, (i.e.P), or a reduction in the protection against pattern vote attacks (in comparison with the basic partition described in Section 4.1)[1].

The first solution creates multiple distinct partitions of the set of contests, each partition containing a set of subsets of the contests, so if we have 5 contests (c_i), we may have a (p, n, c) partitioning, where p is the number of distinct partitions, n is the number of groups of contests, and c is the number of contests. The size of each grouping is between 1 and c, and will need to be small enough so that patterns are not visible, but large enough to permit statistical analysis. For example we could specify a (3,2,5) partition (with 2 subsets, and 5 contests) c_1, c_2, c_3, c_4, c_5 and a second partition of c_2, c_3, c_4, c_1, c_5, and the third partition of c_1, c_3, c_5, c_2, c_4. Then p groups of D tables are created, with each group consisting of n D tables.

Each D table will have B/p rows in it, since each vote only is mapped to one of the partitions. The corresponding R table will also have B/p rows. The results of any one contest can be read by adding the votes for that contest from each of the R tables containing votes for that contest. For example contest 1 will be read from the first R table in partition 1, the second R table in partition 2 and the first R table in partition 3. This modification reduces the privacy set P because each D table is smaller and the pool of equivalent voters that each clear ballot could have come from is smaller.

The other solution has the same basic setup, except that each D table will still have B rows, each entry in the P table (i.e. each vote) will map to a different row in each of the sets of D tables corresponding to a partition. So each vote will be decrypted multiple times and will end up in all of the R tables. Thus the total count of votes for each contest can be computed separately in each of the partitions, and should be identical to that computed for any other partition (this can easily be checked). In this case the privacy set is not reduced, but since each vote is included in all of the partitions, more information about the connection between a voter's vote in one set of contests and their votes in a separate set of contests is available.

6 Conclusions

Cryptographic voting systems represent a promising family of potential solutions to the problem of creating accurate and secret public elections. Whether such systems will be useful in practice depends on whether they can match the expectations and legal requirements of citizens and election officials. The ability to prevent voter fraud and purchased ballots are some of the requirements that a voting system must support. In this paper we presented a specific version of these problems that applies to several currently proposed voting systems and developed solutions for the PunchScan system that not only meet those requirements, but can actually improve on the performance and integrity of the voting system as well.

7 Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments, and David Chaum, the PunchScan team, Ben Hosp and Poorvi Vora for their open and vibrant discussions and insightful comments.

[1] Note, it is still better then not having partitions at all

References

[AR06] Ben Adida and Ronald L. Rivest. Scratch & vote: self-contained paper-based cryptographic voting. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 29–40, New York, NY, USA, 2006. ACM Press.

[Ben87] Josh Cohen Benaloh. *Verifiable Secret Ballot Elections*. PhD thesis, Yale University, 1987.

[CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *EUROCRYPT '97*, LNCS 1233, pages 103–118. Springer-Verlag, 1997.

[Cha81] David L. Chaum. Untraceable electronic mail, return address, and digital pseudonym. *Communication of ACM*, February 1981.

[Cha04] David Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, pages 38–47, January/February 2004.

[Cha05] David Chaum. Recent results in electronic voting. In *Presentation at Frontiers in Electronic Elections (FEE 2005)*, Milan, Italy, September 2005. ECRYPT and ESORICS.

[CRS05] David Chaum, Peter Y. A. Ryan, and Steve Schneider. A practical voter-verifiable election scheme. In *In Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann, editors, ESORICS, volume 3679 of Lecture Notes in Computer Science*, pages 118–139. Springer, 2005.

[FCS06] Kevin Fisher, Richard Carback, and Alan T. Sherman. Punchscan: Introduction and system definition of a high-integrity election system. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK, June 2006. www.wote2006.org.

[FOO93] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *AUSCRYPT '92*, LNCS 718, pages 244–251. Springer-Verlag, 1993.

[HPS+05] Ben Hosp, Stefan Popoveniuc, Rahul Simha, Jonathan Stanton, and Poorvi Vora. Implementation and evaluation of a cryptographically secure voting system. Available at http://vote.cs.gwu.edu/, December 2005.

[HV06] Ben Hosp and Poorvi Vora. Integrity measures from an information-theoretic model for voting. In *Threat Analyses for Voting System Categories: A Workshop on Rating Voting Methods (VSRW06)*, Washington DC, June 2006. NSF/NIST.

[KHR06] Armen Keteyian, Phil Hirschkorn, and Michael Rey. Florida recount, 2006-style: Large 'undervote' in hot house race raises voting machine concerns. http://www.cbsnews.com/stories/2006/11/11/cbsnews_investigates/main2174376.shtml, November 11 2006.

[Nef01] A. Neff. A verifiable secret shuffle and its application to e-voting. In *8th ACM Conference on Computer and Communications Security*, 2001.

[Oka97] T. Okamoto. Receipt-free electronic voting schemes for large scale elections. In *5th Security Protocols Workshop*, LNCS 1163, pages 125–132. Springer-Verlag, 1997.

[oVSC06] Bureau of Voting Systems Certification. Parallel test summary report for sarasota county, fl november 7, 2006 general election using election systems and software, inc. unity version 4.5, version 2. http://election.dos.state.fl.us/pdf/parallelTestSumReprt12-18-06.pdf, December 18 2006.

[PH06] Stefan Popoveniuc and Ben Hosp. An introduction to PunchScan. In *IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Robinson College, Cambridge UK, June 2006. Also GWU-CS Technical Report.

[Pro01] Caltech-MIT Voting Technology Project. Voting what is, what could be. http://www.vote.caltech.edu/, July 2001.

[Riv06] Ronald L. Rivest. The ThreeBallot voting system. http://theory.lcs.mit.edu/~rivest/, October 2006.

[Rja03] Zuzana Rjaskova. *Electronic Voting Schemes*. MSc thesis, Comenius University, Bratislava, 2003.

[SK94] Kazue Sako and Joe Kilian. Secure voting using partially compatible homomorphisms. In *Advances in Cryptology - CRYPTO'94*, pages 411–424, 1994.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme: a practical solution to the implementation of a voting booth. In *Advances in Cryptology - EUROCRYPT'95*, 1995.

[VI03] VoteHere Inc. Documentation. http://www.votehere.net/vhti/documentation/verifiable_e-voting.pdf, 2003.

[Vor05] Poorvi L. Vora. David chaum's voter verification using encrypted paper receipts, 2005.