Secure Internet Voting with Code Sheets

Jörg Helbach[1] and Jörg Schwenk[2]

[1] Sprint Sanierung GmbH, D-51061 Köln
joerg.helbach@sprint.de
[2] Ruhr-University Bochum, D-44780 Bochum
joerg.schwenk@rub.de

Abstract. Malware on Personal Computers is a major security issue today. This fact implies that all solutions intended to secure Internet-based voting have to be re-evaluated under the assumption that a local malware application is capable of controlling the interface between user and PC. We propose to use paper-based code sheets, originally introduced by Chaum, to overcome this problem, and for the first time give a security analysis of this solution. We show that a modified, 3-step-scheme, can be considered secure against local malware attacks. Our scheme could then particulary be used to held shareholder elections or votes in an associaton over the Internet.

1 Introduction

Since 2004, malware has evolved from the playground of script kiddies to a valuable tool for cyber criminals. Trojan horses have been found in the wild by AV companies that control the complete input/output of browser based online-banking applications, even if there is an authentic SSL connection with the bank [Gri06][Emi06]. According to Vinton Cerf, co-developer of TCP/IP and architect of the Internet, the problem is immense: "Of the 600 million computers currently on the Internet, between 100 and 150 million were already part of these botnets" [Web07]. Against this background, it seems naïve to propose to perform elections over the Internet. However, political elections in Estonia were held over the Internet (fortunately only about 3% of all voters voted this way, which resulted in statements from officials that this kind of voting was secure) [Est07] and a new EU directive states that shareholders must be allowed to vote over the Internet [Zyp07]. We propose a solution introduced originally as "SureVote" by David Chaum [Cha01], and, up to our knowledge, for the first time analyze its security. In this scheme, the voting agency prints paper sheets as can be seen in table 1, which contain the list of voting alternatives (e.g. names of candidates or yes/no) together with two random numbers, which are chosen separately for each sheet and each voting alternative on this sheet. The first number, which we call Voting Transaction Number (or Voting TAN for short), must be entered into the web form, and a successful cast of the vote is communicated to the voter by displaying the second number, thus called Confirmation TAN. Our security model completely differs from "classical" security models for electronic voting

A. Alkassar and M. Volkamer (Eds.): VOTE-ID 2007, LNCS 4896, pp. 166–177, 2007.

in that we consider the voting authority to be trustworthy, and the personal computing devices of the voters (i.e. their PCs) to be insecure. Thus the voter has no possibility to perform cryptographic computations, because he simply can not trust the outcome of these computations.

2 Related Work

2.1 Voting Requirements

Generally, all elections have to be at least free, universal, secret and equal. In some states elections also must be direct. Many research has been done to adopt voting systems to this properties. Summarized, there are five requirements and threats a voting system has to deal with:

- **Universal election.** A first threat is to prevent the voters from voting. For that purpose an attacker can e.g. use a denial of service attack. Either the voting servers or clients are prevented to provide their services or the communication channel is made unavailable.
- **Direct election.** The second threat is that the ballot is modified before cast. E.g. a Trojan horse or a man in the middle could invalidate a ballot by adding too many additional votes.
- **Secret election.** Third, the loss of the voter's privacy is possible, i.e. the secrecy of the ballot is not guaranteed. As every election has to be free, universal, equal and secret, the basic principles of elections would be violated.
- **Equal election.** Another threat is that a voter could try to cast more than only one ballot.
- **Free election.** The last threat is vote buying, selling and trading. That is, if a voter sells his voting decision to an attacker. Therefore, a voter must not be able to prove his decision to the vote buyer.

An issue that all those threats have in common is scaling, because it is very easy to use computers for automated attacks. If a voting system is vulnerable to one or more of those threats, it mostly fails on a large scale.

The voting scheme based on code sheets deals with all those issues, except of DoS attacks. Vote selling is a problem that is difficult to solve with a purely paper-based solution, because a voter may simply sell the code sheet. A partial solution to this problem is to use vote updating to receive receipt-freeness. However, vote selling is tolerable or even allowed in some non-political elections, e.g. the election of a CEO by shareholders of a company.

2.2 Voting Technologies

In general, all proposed voting systems use different technologies to meet those requirements.

Homomorphous encryption. The first approach uses homomorphous encryption, so that $e(x_1) + e(x_2) = e(x_1 + x_2)$ where x_1, x_2 are ballots and $e(x)$ is an

encrypted ballot. A single ballot is never decrypted even when computing the tally. A voting system, which uses homomorphous encryption, is e.g. CyberVote[1] [Cyb03]. The voting authorities publish all encrypted ballots on a bulletin board. Additionally all ballots are added. Then just the result of the addition is decrypted, so everyone can prove that the tallying and the addition were correct. Using anonymous channels this voting scheme can receive receipt-freeness, universal verifiability, privacy and uncoercibility.

Mix-Nets. The second approach, using Mix-Nets, permutes the messages on the communication channel, so that the order of the incoming ballots is changed to guarantee the anonymity of the voters. Mix-Nets are often used in combination with homomorphous encryption systems.

Blind signatures. The third type of a voting system uses (blind) signatures. Here the voter sends his encrypted and signed ballot to a canvasser, who checks if the voter is eligible. In this case, the canvasser signs the ballot himself and sends it back to the voter. Now the voter can remove his own signature, so that he gets an encrypted ballot, which was signed by a canvasser. The voter can now send that ballot to the urn, which can decrypt and count it. This approach was firstly introduced 1982 by David Chaum [Cha82]. A voting system used in practice is e.g. Polyas[2] [Mic06]. Based on this concept, many election protocols have been proposed. A very important voting scheme based on blind signatures is the system Fujioka, Okamoto and Ohta proposed in 1992 [Fuj92]. In this scheme the counter publishes the encrypted ballots, so every voter can check if his ballot is on the list. The problem here is that this voting scheme is not receipt-free, i.e. the voter can prove his decision to a third-party and thus can sell his vote.

Paper based voting systems for general elections. Lately, some concerns have been raised about the acceptance of voting systems based on cryptographic functions that will not be understood by a majority of voters, or whose implementation cannot be verified publicly. As a consequence, we have seen a renaissance of paper based voting schemes, which cleverly combine simplicity and security. An important property of an election system is, whether the voter can verify, that his ballot had been computed correctly in the tally. In 2004 Chaum, Ryan and Schneider proposed a voter-verifiable voting scheme, Prêt-a-voter [Cha04]. In that voting scheme the ballot is paper-based and consists of two columns. On the left column the candidates are listed in a random order. On the right one the corresponding bubbles are printed, where the voter can make his choice. Additionally on the right column an encrypted code is printed, which contains the information how to reconstruct the order of the candidates, without having the left column. In an election process a voter gets two paper-ballots. The voter selects randomly one of those ballots. The voter checks this ballot, if the random

[1] Several elections were conducted with a prototype of the CyberVote voting system. All cast ballots were no real votes, but just test ballots. An improved variant of CyberVote is currently commercialized.

[2] Polyas has been used for the chairmanship elections in 2004, 2005 and 2006 of the German association of computer science. About 12.000 real votes were cast.

order of candidates on the left side suits the encrypted code on the right. This checks can be done in different ways, e.g. a dummy vote or giving the encrypted code to a canvasser, who returns the order of candidates[3]. If the ballot is correct, one can assume that the second ballot is correct, too, and the first paper ballot can be destroyed. On the second ballot the voter marks his choice, detaches the left column and destroys it. The remaining ballot is fed into an optical reader, which reads the position of the voters cross and the encrypted code.

In 2006 Adida and Rivest enhanced Prêt-a-voter [Adi06]. On the right column the encrypted code is exchanged by a 2D-barcode, which contains the order of candidates, and a scratch surface. The voter makes his choice as described above. After the left column has been detached and destroyed, the voter presents the remaining ballot to a canvasser, who checks and detaches the scratch surface. Then the ballot is scanned and published on a bulletin board, so the voter can check, if his ballot has been computed in the tally.

Another paper based voting system is Rivest's ThreeBallot Voting System [Riv06]. This system doesn't use any cryptography, but only paper ballots. A unique number identifies every ballot. Except this unique ID, all ballots are identical. The voter casts three ballots and gets a copy of one of his ballots as a receipt. The user can choose, which ballot he wants to copy. To vote for a candidate the voter must fill in two of the three bubbles for this candidate, all other candidates must have one bubble filled in, i.e. the voter has to fill in at least one bubble in each row and may not fill in all three bubbles in a row. With these requirements the tallying is quite simple. To receive the election result for a candidate, one just has to count all marked bubbles for this candidate and subtract the total number of voters. As all those voting schemes are paper-based, they cannot be easily adopted to a remote voting system.

2.3 Voting Systems for Internet Elections

Besides those improvements for paper ballots or direct-recording voting machines, which are running in a voting booth inside a polling station, researchers also developed voting systems, which are used completely over the Internet. In the literature these systems are often named Remote Online Voting Systems. Early remote online voting systems were Sensus by Lorrie Cranor [Cra97] and E-VOX by Mark Herschberg [Her97]. Both Sensus and E-VOX are based on the Fujioka, Okamoto, Ohta protocol, mentioned above. Obviously, remote online voting systems have to deal with the same requirements as voting systems in general. Additionally those systems have to ensure the voters' authentication over the Internet, i.e. that the voter is eligible and has not voted before. It seems to be easy to authenticate a voter, but, however, as the election has to be anonymous, too, it is one of the main technical problems to bring authentication and anonymity together. There are different possibilities to authenticate a voter. Authentication can be done by issuing a username and a password to all voters, with what the voters can login into the voting application. Another

[3] For details see [Cha04].

possibility is to authenticate by SmartCard. Certainly, it must be guaranteed, that all polling stations, i.e. the personal computer, are attached to an adequate card reader. Biometric authentication is also possible, but research into biometric authentication showed, that the false rejection rate is currently too high for such a sensitive environment as electronic voting [Hof04]. It is obvious that the voting systems, which use one of those authentication methods, have to deal with the anonymity of the voters. A fourth alternative is the usage of a transaction number. Each voter is issued such a number at random, so anonymity of the voter is given. Other problems using remote online voting systems are the "Secure Platform Problem", introduced by Rivest [Riv02], and denial of service attacks on the voting servers. In 2001 Chaum proposed an approach to deal with malware on the client computers, using code voting [Cha01]. In 2002 Oppliger referred to code voting as code sheets [Opp02]. On every ballot for each candidate is generated a random number. The voter makes his choice by submitting that random number to the voting server. The server responds with a verification number. That voting scheme is not receipt-free, for the voter can prove his vote by giving his code sheet and the verification number to a coercer. To receive receipt-freeness in some voting systems is used vote updating [Vol06][Ram02]. A voter can overwrite his decision as many times as he wants. Only the last vote is computed in the tally.

3 Trust Model

Traditional democratic elections are protected by the direct involvement of many people from different political parties in the organization of the election[4]. This guarantees that no single person or no small group of persons can modify the outcome of the election, as it is often the case in non-democratic elections. However, we must stress the fact that only a relatively small subset of all voters is involved here. A main concern of modern voting schemes is to broaden the group of people that can check the correctness of the voting process [Sch00]. However, regarding remote online voting systems this comes at the cost of performing highly complicated cryptographic computations by each voter. The only tool for this task that is available today is the personal PC of the voter. The main drawback of this highly democratic approach is that today many of the PCs are no longer under the control of the voter. To give just one example: In May 2006, an American botmaster who controlled about 400.000 personal computers was sentenced to 57 months in prison [Reg06]. This single person could therefore have controlled nearly half a million votes. Our trust model is therefore based on the assumption that the voter's PC is untrusted. We assume the existence of a trusted voting centre that issues printed code sheets and which is controlled by a democratic selection of people from all political groups. That is, all printed code sheets are correct and distributed to all eligible voters. The voting servers and databases are reliable, secure and trustworthy. We further assume, that the tallying is trustworthy, i.e. the voting authority doesn't add or delete any valid

[4] Separation of duties (SoD).

Table 1. Printed code sheet

Voting TAN	Candidate	Confirmation TAN
738747987	Ronald Reagan	332676873
983293774	Bill Clinton	676476488
192851911	Will Smith	301287123
...	...	...

votes. Another assumption is that vote selling and buying is either not permitted or even allowed as it is in some association elections or elections for shareholders [Zyp07].

4 Voting Scheme

In our voting scheme, we assume that distributing the code sheets by snail mail is relatively secure; i.e. that attacks are possible, but that the percentage of attacked code sheets is negligible compared to the total number of votes. In other words: attacks do not scale! In contrast to this, we do not make any assumptions regarding the security of either the communication channel through the Internet, or on the security of the client PC used for voting. Nevertheless we are able to prove that no attacker is able to perform successful attacks other than DoS.

The voting scheme works as follows:

Phase 1: Setup. In the Setup phase, the trusted voting authority generates the voting TAN lists. For each list and for each candidate, a random voting TAN and a random confirmation TAN is chosen. This data is printed on a code sheet, and stored as a record in the secure database of the voting centre. The addresses of voters are printed on envelopes, and the TAN lists are inserted into the envelopes in a random order. Then the lists are sent by snail mail to the voters. This process can be democratically assured by controlling the permutation of code sheets (e.g. by manually shuffling the lists) and by randomly selecting samples from the outgoing mail to check whether it contains a valid TAN list. (An additional randomization effect could be achieved by exchanging code sheets at "voting parties". This would improve the democratic trust in code sheets. However, it must be guaranteed that the authenticity of code sheets can be easily verified.)

Phase 2: Voting. Each voter opens the voting web page and enters the voting TAN printed left of the chosen candidate's name. As a result, the voting web page displays the confirmation TAN so that the voter can check that he did not make any typing errors. To provide receipt-freeness, even though the voter has a confirmation TAN and his TAN list to prove his decision, vote updating[5] is allowed. That is, multiple casts per VoterID are possible, but only the last vote counts[6]. If the chosen voting TAN is not accepted or the voter gets a

[5] See also [Ram02].

[6] A detailed analysis of multiple casts is given in [Vol06].

false confirmation TAN, he or she should claim to the voting authority. As we assume, that the voting authority is trustworthy and the distributed code sheets are correct, one can conclude that in this case either the voter made a mistake or the voting client is infected.

Vote Updating. Vote updating can be used to minimize the threats from DoS attacks and vote buying. If vote updating is allowed, the voter can be encouraged to vote early (because he can modify his decision later), and thus effects of DoS attacks in the final stage of an election can be minimized. On the other hand, if a voter sells his code sheet, he may keep a copy and try to update his vote very lately. However, even if the vote-seller tries to update his or her vote, he or she is racing with the vote-buyer, who can arrange to be almost certain to win the race, since he can re-perform the update as many as times as needed. We have to assume that the vote-buyer probably has more resources and patience than the vote-seller, and for instance can automate the process of repeatedly sending updates.

Phase 3 (optional): Public verification of the election result. The result of the election can be made publicly verifiable by publishing a list of all cast voting TANs (last vote in case of vote updating) together with the candidate they represent, sorted by voting TAN. Each voter can then check that his vote was counted correctly, and compute the number of votes for each candidate. The drawback of this solution is that it facilitates vote selling because the voter is able to prove what he voted by showing the code sheet to the buyer. However, we assume a scenario, where vote selling is either not permitted or even allowed.

5 Security Analysis

In this section we will show, that our proposed voting scheme is secure and only allows denial of service attacks and vote selling. As mentioned before, there are several threats a voting system has to deal with. For all those threats we analyze, which type of attack is remarkable and if our voting scheme is vulnerable to those attacks.

5.1 Communication Model

In our model, the human voter directly communicates with the voting server. To protect this communication against active and passive adversaries, we use ideal cryptographic functionality. In this model, integrity and confidentiality of voting ballots are not protected by encryption and/or digital signatures, but by sending a valid handle to a database entry:

- When the Voting TAN is sent over the Internet, the confidentiality of the candidate's name is protected by the fact that the TAN is only a randomly chosen handle to this name, known only to the central database and to the voter (via the code sheet).

– The integrity of the Voting TAN is protected by the fact that the TANs are sparsely distributed in a large number space. Any attempt to fake a Voting TAN will thus with high probability result in a number not belonging to this sparse set.

5.2 Attacks by a Passive Adversary

A passive adversary is only able to attack the secrecy of an election. In our model, the passive adversary can observe the Voting TAN entered into the web browser, and the confirmation TAN sent by the voting server. Since he by assumption has no access to neither Code Sheet nor to the database of the voting server, and since both TAN numbers were chosen at random, he can not do better than simply guess the vote. This guess may be biased by the result of the election, which does not give any additional information about the individual vote.

5.3 Attacks by an Active Adversary

In addition to the abilities of the passive adversary, an active adversary may insert any message into the communication channel between the human voter and the voting server. This includes sending TAN numbers, which have been observed or randomly chosen, in both directions, and error messages of any kind. Error messages are a means to trigger some attacks based on social engineering, as explained later.

Disenfranchisement/DoS. In our context disenfranchisement is preventing the voter from participating in the election. For this, an attacker can perform a denial of service attack (DoS). In our context a DoS is the attempt to make the voting system unavailable to the voters. Our proposed voting scheme cannot deal with general DoS. However, we make the assumption that DoS attacks based on local malware can only prevent a fixed (small) fraction of voters from voting. The advantage of our solution is that denial of service attacks could be detected.

Modifying a ballot: Voting for a specific candidate. In this attack the adversary tries to send Voting TANs to the server that are associated with a specific candidate. To this purpose, he can guess or steal Voting TANs.

– If he guesses a Voting TAN, he can not vote for a specific candidate, since the valid Voting TANs are equally distributed over all candidates.
– If he steals a Voting TAN, he gets no information about which candidate this TAN is associated with, since the Voting TAN was chosen randomly. This voting TAN can only be used for the candidate chosen by the voter.

Modifying a ballot: Averaging attack. In this attack the adversary simply guesses a Voting TAN, and sends it to the voting server. If his guess was correct, the voting server answers with a Confirmation TAN. Since valid TANs are equally distributed over all candidates, this attack will have an averaging effect amongst all candidates: favorite candidates will get less votes, outsider candidates will

get more votes. Since vote updating is allowed, this attack will proceed in two stages: (1) In the first stage, which starts at the beginning of the voting period, the adversary guesses Voting TANs, and stores successful guesses in a database. (2) Towards the end of the voting period, all valid Voting TANs stored in the database will be cast again, to overwrite the votes of legitimate users. To make the voting system resistant against this kind of attacks, the averaging effect must be minimized such that it has no effect on the voting outcome. The following parameters are important here:

- The number n of guessed Voting TANs the adversary is able to transmit to the voting server during the voting period. This number depends on several factors: the number of PCs controlled by the adversary, the number of invalid votes accepted per PC/browser, ...
- The number m of digits of the Voting TAN.
- The number k of candidates.
- The number v of voters.

To guarantee that this attack has no effect on the outcome of the election, the number α of votes per candidate that can be guessed must be negligible.

$$\alpha = \frac{n \cdot v \cdot k}{k \cdot 10^m} = \frac{n \cdot v}{10^m}$$

E.g. if $n = 10^6$ and $v = 10^8$, then choosing $m = 15$ would guarantee that less than one vote per candidate can be generated this way.

Modifying a ballot: Vote-for-the-outsider attack. In this attack, the adversary substitutes the Confirmation TAN repeatedly by an error message. The voter is thus tricked into believing that there is a printing error in the Voting TAN of his favorite candidate. A possible reaction is to test another voting TAN (since vote updating is allowed). It may be argued that the voter will not use the TAN of the biggest competitor of his candidate, but an outsider. This is a very severe attack, since its impact only depends on the number of PCs controlled by the adversary, and cannot be decreased by e.g. increasing the length of the Voting TAN. The attack can be triggered by other events, e.g. a candidate changing his political direction from left to right (or vice versa), to win new votes, and inject old votes recorded by malware.

Table 2. Printed code sheet with Finalization TAN

Voting TAN	Candidate	Confirmation TAN	Finalization TAN
738747987	Ronald Reagan	332676873	442367810
983293774	Bill Clinton	676476488	123456789
192851911	Will Smith	301287123	520172861
...	...	...	...

A solution to this problem is depicted in table 2. Each vote consists of two parts: the voting TAN and the finalization TAN. The vote will only be counted if both parts arrive at the server. If we assume that no user enters the finalization TAN if he hasn't seen the confirmation TAN before, we can prove the security of our scheme. The problem we are facing resembles the classical (unsolvable) "two army problem" from communication theory [AEH75][Gra78]: Two parties (here the voter and the voting server) try to communicate through a channel controlled by the adversary. They can never reach a state where they have the same information, because if a message arrives at the receiver, the sender does not know if the message was successfully transmitted. Our solution is along the lines of three-way-handshakes used in computer communications (e.g. TCP handshake).

We assume that a voter never would enter the finalization TAN, if the voting server diesn't respond with the correct confirmation TAN according to the submitted voting TAN. Therefore, in the counting phase a ballot could have two possible patterns: First, both the voting TAN and the finalization TAN were submitted correct to the voting server. In this case the vote is valid, and hence is considered in the tally. This vote only could be overwritten by a valid voting TAN, finalization TAN combination, which belongs to the same code sheet. Second, the voting TAN was submitted correct to the voting server, but the finalization TAN wasn't submitted. In this case the voting client could have faced an attempt to defraud or an DoS attack. Those votes are not computed in the tally, but published on a special bulletin board. The voters are challenged to check, if his or her voting TAN is published on that particular board and hence should recast their vote using another voting client.

Assuming that the vote for the outsider attack is successful, one can identify different voting patterns at the voting server regarding the last submitted voting and finalization TANs. These patterns could be:

- V-TAN-outsider, V-TAN-favorite, V-TAN-outsider + F-TAN-outsider
- V-TAN-outsider + F-TAN-outsider, V-TAN-favorite, V-TAN-outsider + F-TAN-outsider
- V-TAN-outsider + F-TAN-outsider, V-TAN-favorite + F-TAN-favorite, V-TAN-outsider + F-TAN-outsider

Votes with those patterns are also published on the bulletin board, with challenging the voters to check their vote.

5.4 Vote Selling, Buying and Trading

Vote selling is a major problem in political elections. In our voting scheme we assume an elections where vote selling is legal, e.g. votes for a CEO of a stock company. In this case shareholders may legally trade their voting rights, so there is no reason to prevent this in a code based scheme.

However, if vote selling is prohibited, and because Internet voting can make it easy to buy a large scale of votes by automating the process, it is a demanding

task for a voting scheme, to eliminate the possibility of vote buying. In our proposed voting scheme a voter has different possibilities to sell his decision:

- He can sell his code sheet to an attacker. This process is not scalable, because the code sheets are sent by snail mail to the voters. (This is comparable to sell a vote that is cast today in political elections by snail mail.) Furthermore, as the buyer cannot be sure that the voter didn't make a copy of the TAN list to vote at a later moment using vote updating, this possibility is very unattractive, because it results in a race condition.
- If public verification is used, a proof for correct vote selling may consist of a voting TAN sent to the buyer before the results of the election are published on the public bulletin board.
- As a possible countermeasure one could propose to include for each candidate A a fixed number i of invalid pairs (voting TAN, A) in the election result list, and to print on the code sheet one randomly selected pair for each candidate. However, in this case the buyer could simply request two voting TANs!

To summarize: Without public verification, code sheet based voting schemes are as vulnerable to vote selling as snail mail voting is today. With public verification, vote selling is as easy as sending a voting TAN to the buyer!

6 Conclusion

We proposed a very simple and small voting scheme, which is not vulnerable to most of the threats introduced by malware. This is achieved by using a 3-step-process where random TANs are exchanged between voter and voting server. An adversary can influence the absolute number of cast votes, but not their relative distribution.

Legal problems are another issue. How the case of even small DoS attacks can be handled legally is outside the scope of this paper, and vote selling may become a problem if code sheets are used on large scale.

References

[AEH75] Akkoyunlu, E.A., Ekanadham, K., Huber, R.V.: Some Constraints and Tradeoffs in the Design of Network Communications. ACM SIGOPS Operating Systems Review 9(5), 67–74 (1975)

[Adi06] Adida, B., Rivest, R.: Scratch & Vote. In: Proceedings of the 5^{th} workshop on privacy in the electronic society, pp. 29–40 (2006)

[Cha82] Chaum, D.: Blind signatures for untraceable payments. In: Advances in cryptology, pp. 199–203 (1982)

[Cha01] Chaum, D.: Sure Vote: Technical Overview. In: Proceedings of the workshop on trustworthy elections (WOTE 2001), presentation slides (2001), `http://www.vote.caltech.edu/wote01/pdfs/surevote.pdf`

[Cha04] Chaum, D., Ryan, P., Schneider, S.: A Practical, voter-verifiable Election Scheme, Technical Report 880, School of computing science, Newcastle university (2004)

[Cra97] Cranor, L.: Sensus: A Security-Conscious Electronic Polling System for the Internet. In: Cranor, L. (ed.) Proceedings of the Hawai'i International Conference on System Sciences (1997)

[Cyb03] Website of the CyberVote Project, http://www.eucybervote.org

[Emi06] Emigh, A.: The Crimeware Landscape: Malware, Phishing, Identity Theft and Beyond. A Joint Report of the US Department of Homeland Security–SRI International Identity Theft Technology Council and the Anti-Phishing Working Group

[Est07] Estonia to hold world's first Internet election, http://www.pcpro.co.uk/news/105714/estonia-to-hold-worlds-first-internet-election.html

[Fuj92] Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, Springer, Heidelberg (1993)

[Gra78] Gray, J.: Operating Systems. LNCS, vol. 60, pp. 393–481. Springer, Heidelberg (1978)

[Gri06] Grimes, R.: An SSL trojan unmasked, http://www.infoworld.com/article/06/03/03/75970_10OPsecadvise_1.html

[Her97] Herschberg, M.: Secure Electronic Voting Over the World Wide Web, Master Thesis, at the Massachusetts Institute of Technology (1997), http://theory.lcs.mit.edu/∼cis/voting/herschberg-thesis/index.html

[Hof04] Hof, S.: E-Voting an Biometric Systems? In: Proceedings Electronic Voting in Europe Technology, Law, Politics and Society. LNI P-47, pp. 63–72 (2004)

[Mic06] Polyas Website, Micromata Objects GmbH, http://www.polyas.de

[Opp02] Oppliger, R.: How to Address the Secure Platform Problem for Remote Internet Voting. In: Proceedings 5^th Conf. Security in Information Systems (SIS 2002), vdf Hochschulverlag, pp. 153–173 (2002), http://www.ifi.unizh.ch/~oppliger/Docs/sis_2002.pdf

[Ram02] Wu, C., Sankaranarayana, R.: Internet Voting: Concerns and solutions. In: International Symposium on Cyber Worlds: Theories and Practices (2002)

[Reg06] http://www.theregister.co.uk/2006/05/09/botnet_master_ancheta_jailed/

[Riv02] Rivest, R.: Electronic voting. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 243–268. Springer, Heidelberg (2002)

[Riv06] Rivest, R.: The ThreeBallot Voting System. unpublished draft, version 10/1/06, comments appreciated, http://theory.lcs.mit.edu/∼rivest/Rivest-TheThreeBallotVotingSystem.pdf

[Sch00] Scheier, B.: Voting and Technology. Crypto-Gram (2000), http://www.schneier.com/crypto-gram-0012.html

[Vol06] Volkamer, M., Grimm, R.: Multiple Casts in Online Voting: Analyzing Chances. In: Electronic Voting (2006)

[Web07] Weber, T.: Criminals 'may overwhelm the web, http://news.bbc.co.uk/1/hi/business/6298641.stm

[Zyp07] Zypries: Virtual general meetings. throughout Europe enhance the rights of shareholders, http://www.bmj.de/files/-/1739/Press%20Release%20in%20english.pdf