How to Address the Secure Platform Problem for Remote Internet Voting

Rolf Oppliger
eSECURITY Technologies Rolf Oppliger (www.esecurity.ch)
Beethovenstrasse 10, CH-3073 Gümligen
rolf.oppliger@esecurity.ch

1 Abstract

There are many possibilities to implement Internet voting. The casting of ballots at private sites where the voter (or a third party acting on behalf of the voter) administers and controls the voting client, platform, and operating environment is particularly interesting and challenging. This possibility is commonly referred to as remote Internet voting. The most important security problem related to remote Internet voting is the secure platform problem (i.e., the problem that malicious software may attack and modify the remote Internet voting client to change the voter's ballot in some meaningful way). This paper overviews and discusses some technical approaches to address the secure platform problem. It concludes that the combined use of code sheets and test urnes provides a reasonable approach that can be used to achieve a level of security that is comparable to other forms of absentee balloting, such as, for example, voting by postal mail. The open questions related to the use of code sheets are usability and user acceptance of the modified voting behavior.

2 Introduction

Elections and votes are at the heart of all democracies. In fact, they are important bulding blocks and processes for the proper operation of a democratically legitimated government:

- Elections are used to empower politicians to speak for the people (i.e., they are used for delegation);
- Votes are used to query the political will of the people (i.e., they are used to challenge political decisions).

In either case, registered voters must be provided with ballots and voters must cast their ballots in some defined way.

In the literature, the term electronic voting (e-voting) is used to refer to elections and votes that are supported by electronic means. Independent from this term (i.e., e-voting), the idea of using electronic means to support elections and votes has attracted many people in the past. For example, in June 1869, Thomas A. Edison received U.S. patent 90,646 for an „Electric Vote-Recorder“ intended for use in Congress. Since then, various systems directly or indirectly related to e-voting have been invented, approved, implemented, partly revised, or rejected. Some of these systems have been granted patents,[1] whereas others have been protected with other means of intellectual property protection (e.g., trade secrets).

With the deployment and wide proliferation of the Internet, its use for e-voting has been proposed by many people as a way to make voting more convenient and—as it is hoped—to increase participation in public elections and votes. In this paper, the term Internet voting is used to refer to any election or voting process that enables voters to cast their ballots over the Internet in some way or another. This basically means that the ballots must be represented electronically, and that the electronic ballots must be transmitted to election officials using the Internet as a transport medium. For example, in the U.S. the Arizona Democratic Party used Internet voting in March 2000 for its Presidential Preference Primary.[2] The election involved several thousands of voters and was an official election in the sense that the result was binding. The e-voting system, however, was neither public nor certified by the State of Arizona (since the election was internal to the Democratic Party). For such a system to be used for public elections or votes, it would have to be certified by the state where it is being used. As of this writing, there is no state that has officially certified such a system.

3 Internet Voting

There are many possibilities to implement Internet voting. For example, depending on the places where the ballots are casted and who administers and actually controls the voting clients, platforms, and operating environments, poll-site Internet voting and remote Internet voting are usually distinguished.

---

[1] A list of U.S. patents related to e-voting can be found, for example, at http://www.safevote.com/patents.

[2] The company election.com was appointed to conduct the election. Further information can be found on the company's home page at http://election.com.

- Poll-site Internet voting refers to the casting of ballots inside official polling places at sites where election officials administer and fully control the voting clients, platforms, and operating environments.[3]
- Contrary to that, remote Internet voting refers to the casting of ballots at private sites (e.g., home, office, school, ...) where the voter (or a third party acting on behalf of the voter) administers and controls the the voting client, platform, and operating environment.

Considering the media attention that has focused on the prospect of using the Internet to vote, it is not suprising that the terms „Internet voting" and „remote Internet voting" are being used synonymously in the popular press. As discussed later, however, it makes a lot of sense to cleanly distinguish between the two terms.

A third possibility offers an intermediate step between poll-site Internet voting and remote Internet voting.

- Kiosk voting refers to the casting of ballots outside official polling places at sites that are publicly accessible (e.g., shopping malls, post offices, libraries, schools, ...). The voting clients and their platforms are administered and controlled by election officials, but the operating environments can not be fully controlled by them. Where necessary and appropriate, however, surveillance and monitoring technologies may be used to remotely control the operating environment. Note that kiosk voting is conceptually similar to the use of automatic teller machines (ATMs) in the financial industry.[4]

As discussed below, the three possibilities to implement Internet voting have specific security properties and implications.

4 Security Requirements

There are many investigations and studies that elaborate on the security of Internet voting in general, and remote Internet voting in particular (e.g., [Cal00,IPI01,Rub01]). The results unanimously agree that security (including privacy and reliability) is among the most

---

[3] In some references (e.g., [Cal00]), a distinction is made between poll-site Internet voting where a precinct polling place must be used, and poll-site Internet voting where any official polling place may be used. This distinction is not made in this paper and both possibilities are collectively referred to as poll-site Internet voting.

[4] In this analogy, poll-site Internet voting is conceptually similar to physically visiting a bank and remote Internet voting is conceptually similar to Internet banking.

important engineering considerations for Internet voting to be successful in the first place. The current paper ballot systems set a standard that is adopted as the baseline for Internet voting. They represent certain tradeoffs between voter convenience and protection against fraud and abuse. It is generally required that elections and votes conducted over the Internet are at least as secure as the current paper ballot systems. If a state allowed voting by postal mail, however, this mechanism sets the security standard for Internet voting.

Also, it is essential that an Internet voting system provides some evidence that it is immune from attacks that could affect the outcome of an election or vote. It is not sufficient to argue that a specific attack is unlikely, or even very unlikely, to happen. An election or vote would be an extremely tempting target for any motivated party (e.g., a hacker group, group of partisans, foreign government, ...). Such an attack would be a political and public relations disaster; or worse, if successful and undetected, compromise the results of the election or vote. It must be presumed therefore, that if a specific attack is possible, it will happen sooner or later. Even before anything happens, people will publicly criticize Internet voting systems that are subject to specific attacks. There is some risk that the public would loose confidence and trust in the system (maybe even before anything happens at all).[5]

When talking about security, there are several requirements that must be considered with care. The following list is not intended to be complete and comprehensive:

- Completeness and soundness of the voting protocol;
- Correctness of the results;
- Authenticity of both the voter (or the voting client acting on behalf of the voter, respectively) and the voting server;
- Secrecy of the ballots (including, for example, anonymity of the voter);
- Integrity of the ballots (including, for example, protection against malicious software[6]);

---

[5] Note that confidence and trust are properties that are hard and time-consuming to establish, but they can be lost very rapdily.

[6] Malicious software is software that is deliberately designed to do harmful things that the user neither wants nor expects, and to hide the harmful action or perform it so

- Non-duplication of the ballots;
- Availability and reliability of the voting process (including, for example, protection against denial-of-service attacks).

Some security requirements are complementary and don't interact with each other (e.g., integrity and non-duplication of the ballots). Other security requirements, however, are (or at least seem to be) contradictory. For example, one way to attest the correctness of a voting process is auditability, meaning that the entire voting process can be audited in some reasonable way. Auditability, however, sometimes also contradicts to the secrecy of the ballots. In fact, there is a lot of research going on in the cryptographic community to address this apparent contradiction and to guarantee ballot secrecy and the correctness of the results at the same time [Sch00]. Most of this research elaborates on schemes and protocols for secure multi-party computation (e.g., [Hir01]).

Against this background, it is important to note that the security requirements of e-voting are fundamentally different and more difficult to satisfy than the ones of electronic commerce (e-commerce). In e-commerce, financial transactions are performed online, but there is always a separate offline process for checking them and for correcting any errors detected. This is not, and cannot be, the case for e-voting.[7] Therefore, the fundamental security emphasis in e-voting must be prevention of fraud and error, with no reliance on any possibility of after-the-fact correction. This is a much more stringent requirement than is generally necessary today for financial and e-commerce transactions.

Most security requirements of Internet voting can be addressed with existing technologies, mechanisms, and services (e.g., [Opp02a, Opp02b]). For example, the authenticity of the voter and the voting server can be addressed with public key certificates. Similarly, the secrecy and integrity of the ballots can be addressed with the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocol. It is,

---

quickly that it cannot be stopped. Malicious code is also known as malware or vandalware. These terms, however, are not used in this paper. Malicious software is usually distributed to computer systems through a variety of mechanisms known as computer viruses, worms, Trojan horses, back doors, trapdoors, or logic bombs.

[7] This is because it must be made impossible to sell votes. Note that if a voter received a proof for his or her actual vote (i.e., the ballot he or she actually casted), he or she could sell it and large-scale vote selling and buying would become a problem.

however, important to note that the use of the SSL/TLS protocol protects the secrecy and integrity of the ballots only during their transmission over the Internet. The ballots are not automatically protected at the client or server side. In fact, additional security technologies, mechanisms, and services are required to protect the secrecy and integrity of the ballots before and after they are transmitted over the Internet. Consequently, there are some additional risks for the secrecy of the ballots (i.e., privacy risks) related to the use of spyware[8] (e.g., in the home setting) and remote system administration tools (e.g., in the institutional setting). Fortunately, the use of code sheets as recommended in this paper protects the voter against these additional privacy risks.

From a security point of view, the three possibilities to implement Internet voting (i.e., poll-site Internet voting, kiosk voting, and remote Internet voting) have specific security properties and implications. Since election officials control the voting client, platform, and operating environment in poll-site Internet voting, managing the security of such a system seems feasible. Similarly, in the case of kiosk voting, the voting client and its platform are under the control of election officials and can be secured accordingly. Furthermore, the operational environment can be modified as needed and monitored to address security and privacy concerns (e.g., to prevent coercion or other forms of intervention). Consequently, most of the security problems related to kiosk voting could, at least in principle, be resolved through extensions of existing technologies. Contrary to poll-site Internet voting and kiosk voting, however, remote Internet voting still poses substantial security problems and entirely new challenges. Without official control of the voting client and its platform, there are many ways to use malicious software to manipulate a voting process and its results. Against this background, the integrity of the ballots in general, and protection against malicious software in particular, are among the most important security requirements for remote Internet voting to be used on a large scale. Ronald L. Rivest has coined the term „secure platform problem“ to refer to the problem of protecting an inherently insecure platform against malicious software and corresponding attacks [Riv01].

---

[8] Spyware is software that can be used by one user to spy on the activities of another user (on the same or-even more importantly-on another system). A famous spyware is, for example, Backorifice 2000 (BO2K). Further information about BO2K and its source code can be found at http://www.bo2k.com.

It is widely believed that the secure platform problem is the Achilles heel of any remote Internet voting process and system. For example, [IPI01] argues that „remote Internet voting systems pose significant risk to the integrity of the voting process and should not be fielded for use in public elections until substantial technical and social science issues are addressed.“ Similar arguments can be found in [Cal00]. In fact, most relevant investigations and studies conclude

- That the environment that remote Internet voting operates in creates some unique security concerns;
- That currently available client software is far too vulnerable to be used for remote Internet voting;
- That further research is required to come up with security technologies that will eventually solve the problem.

Against this background, most security experts argue that poll-site Internet voting and kiosk voting are feasible in the mid term, whereas remote Internet voting is not feasible. Again referring to [IPI91], „current and near-term technologies are inadequate to address these risks.“ Instead, it is often argued that „any use of the Internet for voting purposes should be phased-in gradually,“ and that Internet voting „would be best served by a strategy of evolutionary rather than revolutionary change“ [Cal00]. This basically means that one should start with poll-site Internet voting systems (phase 1), then move to kiosk voting systems (phase 2), until one finally goes to remote Internet voting systems (phase 3).

There are a couple of proposals that elaborate on how to implement poll-site Internet voting in phase 1:

- In a short note about Internet voting,[9] Bruce Schneier suggested the use of an ATM-style computer voting machine that is physically located at poll-site and that also prints out paper ballots. The voter must check his or her paper ballot for accuracy, and drop it into a sealed ballot box. The voting machine provides the tally, but the paper ballots are still the official votes that could eventually be used for recounts.
- Similarly, a group of researchers from the California Institute of Technology (CalTech), the Massachusetts Institute of Technology (MIT), and Compaq proposed a modular architecture to

[9] The note entitled „Voting and Technology“ can be found in the Crypto-Gram Newsletter of December 15, 2000. An online version of this newsletter can be found at http://www.counterpane.com/crypto-gram-0012.html.

implement poll-site Internet voting in the near-term [BJR01]. The basic idea is that election officials distribute preprepared and empty electronic ballots (so-called „frogs“) to legitimate voters,[10] to have these people remotely generate their vote, and to have the voters cast their votes (i.e., deposit their frogs) at poll-site. The vote casting devices at poll-site are operated and administered by election officials and may provide a reasonable level off security, accordingly. As such, the casted votes can be digitally signed using the devices' private keys. Finally, the frogs provide an audit trail that could eventually be used for disputes and recounts.

Due to the fact that the secure platform problem is known to be hard and difficult to solve, there are also some research and development projects that don't even try to address it. For example, in the FAQ document[11] of the European CyberVote project,[12] the question „Can a virus or Trojan horse attack CyberVote?“ is answered the following way: „Yes, like any other client software in an insecure PC environment. Anti-virus software should be used and strict security guidelines followed to limit the risk of a virus or Trojan horse attack. Secure user interface techniques can be applied to the CyberVote client to prevent Trojan horses.“ Unfortunately, the FAQ document does not elaborate on what is meant with the term „secure user interface techniques.“

In summary, the secure platform problem is known to be hard in the scientific community. The wide belief that is not possible today to implement remote Internet voting in a sufficiently secure way on a large scale, however, only makes sense and applies to an environment that does not provide support for absentee balloting. If a state provides support for absentee balloting (using, for example, voting by postal mail), this line of argumentation does no longer apply. In this sitaution, the security level of any new voting mechanism, such as remote Internet voting, must only be comparable to the security level of voting by postal mail. As discussed in this paper, this level of security seems feasible today.

---

[10] A frog can be represented, for example, by a „dumb“ flash memory card with a lock capability. The architecture, however, is technology-neutral and can be implemented using alternative technologies, as well.

[11] http://www.eucybervote.org/faq_security.html#q35

[12] http://www.eucybervote.org

5 Analysis of the Secure Platform Problem

When talking about remote Internet voting, it is generally assumed that the voting client is an application (program) running on a platform that consists of a personal computer[13] (PC) and a general-purpose operating system, such as Windows or Linux. In a typical setting, the PC is the one the voter uses at home (i.e., in the home setting) or at work (i.e., in the institutional setting). As such, it is administered and operated by the voter or a third party acting on behalf of the voter.

Currently deployed operating systems are open software systems (i.e., they are not software-closed).[14] Users routinely change the systems' functionalities by adding software modules, such as upgrades, patches, device drivers, DLL files, and other extensions acquired from arbitrary sources. The software modules are sometimes added to the operating system as a side-effect of deliberately installing or upgrading application software. In fact, users are often unaware that their operating system has been changed, and certainly have no way of approving or certifying the security and safety of these changes. Similar to legitimate software modules, malicious software can also change an operating system at will.

Application software, such as a Web browser, is often even more openly designed and more casually modified through the addition of software modules (e.g., plug-ins, Java applets, ActiveX controls, JavaScript scripts, ...).[15] In many cases, software modules are downloaded without the user's knowledge as an invisible side-effect of merely visiting a Web site, and yet they have the power to modify the installed software and the behavior of a PC. Many examples of how to misuse this power have been papered in the press. For example, the German Chaos Computer Club demonstrated an ActiveX control that could initiate and queue up an electronic funds transfer using the European version of the Quicken software in 1997. The ActiveX control was written only for demonstration purposes and its developers did not attempt to hide its actions. Consequently, it is

---

[13] In this paper, the term personal computer is meant to include personal digital assistants (PDAs) with corresponding operating systems.

[14] Note that an open software system is not the same as a system that uses open source software.

[15] Sometimes, this type of software is called „mobile code." This term is not used in this paper, mainly because „mobile code" is typically not more mobile than other code. Rather, the characteristic fact of this code is that it is automatically and transparently executed on the client side.

possible and very likely that ActiveX controls can be written and deployed that operate more stealthy and are more dangerous accordingly. The same is true for all programming and scripting langauges in use today.

The easy extensibility of both the operating system and the application software is extremely valuable for the flexibility and adaptability of a PC. It is part of what has allowed the astonishingly fast evolution cycles in the computer industry. The background danger, however, is that any software module can harbor malicious code to attack a PC from the inside. For example, Ken Thompson[16] showed in his 1984 ACM Turing Award lecture that it is very difficult to detect malicious code in an arbitrary piece of software, and that no amount of source-level verification or scrutiny can change this fact.[17] The reason is that malicious code can be introduced at every step in the software production, compilation and execution processes. For example, a modified compiler that autonomously introduces a Trojan horse into compiled software is very difficult to detect (to say the least). Consequently, Thompson concluded that „you can't trust code that you did not totally create yourself“ [Tho84]. Unfortunately, it is not possible to create oneself all code that is neceassry to operate a contemporary PC.

In addition, it is a fundamental theorem of the theory of computation that there can be no general test to decide whether or not a computer system and its software is harboring malicious code.[18] This is unfortunate and as a consequence, commercial virus detection software can detect and neutralize only known computer viruses (they basically scan large amounts of data for known computer virus patterns). They can do nothing or very little about unknown computer viruses. This has to be kept in mind when one talks about operating systems and application sofware that are assumed to be „clean.“

---

[16] Ken Thompson is one of the developers of the UNIX operating system.

[17] This statement does not imply that source code inspection is useless. It only means that source code inspection does not provide a guarantee that the corresponding software does not inlcude malicious code. Source code inspection does provide, however, a general impression about the style of programming and its security and safety properties.

[18] This theorem is a corollary of a theorem claiming that the halting problem is undecidable. This basically means that there is no algorithm that can decide for all possible Turing machines and all possible input strings whether a given Turing machine and input string halts after a finite amount of time.

Taking all of these facts into account, one must admit that the PC as it is used today is a very dangerous platform from which to perform transactions that must be secure. This is true for e-commerce, but it is particularly true for e-voting (the arguments why e-voting is even more critical from a security point of view are given above). If remote Internet voting were permitted from PCs with standard operating systems and standard Web browsers, it would be very simple for a rougue programmer to write malicious software, lure potential voters to download that software (possibly unknowingly), and have the software either spy on the votes, or change them without the voters' knowledge. Consequently, it must be made infeasible—or at least very difficult—to write software that can autonomously do these kinds of things. Some technical approaches are overviewed and discussed next.

6 Technical Approaches to Address the Secure Platform Problem

First of all, it is important to note that the use of cryptography does not help to address—or even solve—the secure platform problem for remote Internet voting. Rather than being a cryptographic problem, the secure platform problem is the problem of how to interface the voter to a cryptographic voting protocol and its implementation. Almost all cryptographic voting protocols assume that a voter has a secure and trusted computing base (i.e., platform) that faithfully executes his or her part of the protocol. More specifically, the platform is assumed to correctly display to the voter his or her intended vote, and correctly submit this vote during the execution of the voting protocol. Consequently, the platform is assumed to act as the voter's trusted agent. To put it into other words: The platform is the voter as far as the voting protocol is concerned [Riv01]. Even if an additional layer of cryptography is added, the problem of how to properly and securely interface the voter to this new layer must be solved.

Contrary to the use cryptography, there are a few technical approaches that can be used to address the secure platform problem for remote Internet voting. The following classification is taken from [Cal00] and is also used in this paper:[19]

- „Clean“ operating system and voting application;

[19] Not all terms are well chosen. Nevertheless, they are used for consistency reasons.

- Special security PC hardware;
- Closed secure devices;
- Secure PC operating systems;
- Code sheets;
- Test ballots;
- Obscurity and complexity.

Unfortunately, not all approaches are technically implementable or enforceable. Their advantages and disadvantages are overviewed and briefly discussed next.

6.1 „Clean“ Operating System and Voting Application

This approach requires that the voter boots his or her PC from a CD-ROM (or a similar read-only medium) that contains an operating system and the voting application client software that are assumed to be „clean.“ The CD-ROM must be designed, producded, and distributed by a trustworthy source (e.g., the state that organizes and manages the voting process).

There are basically two possibilities to design the voting application client:

- The client allows the voter to directly use the Internet to cast his or her vote;
- The application allows the voter to fill out and authenticate a ballot. This ballot is then submitted to an application server at some later point in time (not necessarily using the voting application client software).

In the first case, the CD-ROM must include a sufficiently complete operating system that includes, among other things, all networking software that is required to use the Internet. In the second case, the CD-ROM must include only a small operating system that does not include any networking software. In this case, however, it is necessary to authenticate the ballot after it has been filled out. This authentication requires the computation of a message authentication code (MAC) that can be verified on the server side. MAC computation and verification, in turn, requires a secret key that is shared between the client and the server.

The major advantage of this approach (i.e., using a „clean“ operating system and voting application) is that a certain level of assurance can

be achieved that the software running on the PC used for remote Internet voting is not compromised by malicious software. The level of assurance, however, it hard to quantify, mainly because it is hard to say how „clean“ an operating system and its application software really is. In fact, it happened in the past that software vendors shipped products that had been infected by malicious software.

There are many disadvantages related to this approach. First of all, it is very difficult and challenging to design and produce a CD-ROM from which most PCs in use today can boot from. Some PCs may not even be configured to be bootable from a CD-ROM, and these PCs must be modified at the BIOS level. This is certainly something that goes beyond the technical capabilities of most users. Also, the CD-ROMs must be complete and include all device drivers and software modules that are necessary to use the PC for remote Internet voting. The amount of software primarily depends on which of the two possibilities to design the voting application client (enumerated above) has been chosen. In the first case, for example, the CD-ROM must also include, for example, the drivers for most modems in use today, as well as a full implementation of the TCP/IP protocols. From a voter's point of view, the major disadvantage is related to the fact that he or she must boot the PC before filling out the ballot or casting the vote. This is uncomfortable and in many situations impossible. Also, it is an open question how one would have voters configure their PCs for Internet connectivity in the first case enumerated above (without having the state act as an Internet service provider). Last but not least, it is difficult and not always possible to decide on the server side if a voter has booted his or her PC from an official CD-ROM and if he or she is using the voting application client software from the CD-ROM. Note, for example, that the voting application client can (and is very likely to) be a normal browser.

In summary, the distribution and use of a „clean“ operating system and voting software is a theoretically interesting approach. It is, however, prohibitively difficult and expensive to implement and enforce in practice. As such, it is not considered as a viable solution for the secure platform problem in this paper.

6.2 Special Security PC Hardware

This approach requires a special security PC hardware that is attached to the voter's PC (e.g., through a USB port). The purpose of the hardware is to display the ballot, accept the voter's choices as input, eventually perform some cryptographic computations, and output the result. As such, the voting is done enirely in the special

security PC hardware, and the PC it is attached to is only used as a device to interconnect to the Internet. The important fact about the special security PC hardware is that it is a device that can be made software-closed, meaning that its installed software base cannot be modified (and cannot be attacked by malicious code accordingly).

The major advantage of this approach is the arguably high level of protection against malicious software and corresponding attacks. Since the special security PC hardware can be used only for remote Internet voting, it can be made software-closed and highly secure. This point was already made by the U.S. Institute for Computer Sciences and Technology in 1988 [Sal88]. On the other side, however, the fact that the special security PC hardware is single-purpose also means that it must be provided by the state organizing and managing the vote or a legitimate representative thereof. The major disadvantage of this approach is related to the fact that it is prohibitively expensive to be deployed on a large scale. Consequently, it is not considered as a viable solution for the secure platform problem in this paper.

6.3 Closed Secure Devices

Similar to special security PC hardware, it is possible that special, software-closed, Internet-capable devices may be developed and deplyoed for e-commerce applications. If this were the case, the same devices could also be used for remote Internet voting.

As of this writing, neither are closed secure devices available on a large scale, nor is it likely that such devices will become available and widely deployed soon. Consequently, this approach is not considered as a viable solution in this paper.

6.4 Secure PC Operating Systems

This approach assumes the existence and wide deployment of PC operating systems that are inherently more secure than currently deployed operating systems. Unfortunately, the design, development, implementation, and deployment of a secure PC operating system is very difficult in both theory and practice. There are some research and development projects going on, such as the Trusted Computing Platform Alliance (TCPA[20]) or the Extremely Relibale Operating

[20] http://www.trustedpc.org/home/home.htm

System (EROS[21]). It is, however, not sure whether any of these projects will be successful in the long term, and whether any of the resulting operating systems will be used on a large scale.

Due to their current unavailability, secure PC operating systems are not considered as viable solutions for the secure platform problem in this paper.

6.5 Code Sheets

The basic idea of this approach is to use randomly-looking character strings (representing codes or code numbers) to cast a vote. Consequently, the use of code sheets requires a modified voter behavior. The voter enters a code number istead of „YES“ or „NO“ (in the case of a vote) or a candidate's name (in the case of an election). All code numbers must be distributed on personalized code sheets, and these sheets must be secretly distributed using, for example, postal mail. In either case, the code sheets must be provided outside the reach of the voter's PC (i.e., the PC that is used by the voter to cast his or her vote). If the code sheets were inside the reach of the PC, malicious software could get and use them to change the ballots. Also, the code numbers must be randomly or pseudo-randomly chosen from a sufficiently large set of possible values to make the probability that malicious software can correctly guess a code number arbitrarily small (i.e., negligible).

In the literature, the use of code sheets for voting is sometimes also referred to as „code voting“ [Cha01]. As discussed later, there are many possibilities to implement code voting. In a full implementation, for example, the server may send back a verification number to the voter and the voter can use this number to verify that he or she has casted the vote to an authentic server, and that the vote has been properly registered by the server. In either case, anonymity must be provided by using an additional server system that decodes the ballots and forwards them anonmyously to the actual voting server.

The major advantage of code voting is protection against malicious software without having to boot a PC or install and configure any new hardware or software. Also, the approach is able to protect against the privacy risks mentioned above. If a voter enters a code number (instead of „YES“ or „NO“), anybody using spyware or a remote administration tool is not able to decide whether the voter actually casted a „YES“ or „NO“ (this is not true for a „verification number-

[21] http://www.eros-os.org

only“ implementation). All he or she would see is a code number that looks random. Contrary to that, the major disadvantages are related to the necessity to distribute personalized code sheets on the one hand, and the modified voter behavior on the other hand.

In summary, the use of code sheets is considered as a viable solution for the secure platform problem in this paper. In fact, it is part of the solution that is recommended in this paper.

6.6 Test Ballots

This approach requires that special test ballots are casted from voting clients, and that the proper receipt of these ballots is systematically verified on the server side. If the test ballots are generated in some statistically meaningful way, attacks can be detected and some of these attacks may be caused by malicious software. As such, test ballots can also be seen as an intrusion detection system (IDS) specifically designed and used for remote Internet voting.

The major advantage of this approach is that it works independently from any attack pattern and that it provides a quantitative measure of the size of the attack it detects. Also, it can be used to detect any systematic cause of lost ballots, not just attacks caused by malicious software. Contrary to that, the major disadvantage of this approach is related to the fact that test ballots don't protect against attacks; it only detects them after the fact. Hence they are ideally combined with one (or several) preventative approach(es), such as code sheets.

The use of test urnes is considered as a viable solution for the secure platform problem in this paper. In fact, it is recommended to use them in combination with code sheets.

6.7 Obscurity and Complexity

This approach (also known as „security through obscurity“ in the literature) has a long (but not particularly successful) history in computer security. It basically means that everything related to the voting process (e.g., the format of the electronic ballots, the internals of the voting software, ...) is kept secret prior to the vote and possibly randomly changed during the vote. Also, everything is kept as complex as possible.

The major advantage of this approach is that it makes the writing of malicious software difficult and time-consuming. Contrary to that, a disadvantage is related to the fact that it is difficult if not impossible to specify a lower bound for the amount of time needed to write

malicious software. More worrisome, history has shown that „security through obscurity“ hardly works in practice. More recently, the DVD industry has learned this lesson in an uncomfortable way.[22] Consequently, obscurity and complexity are not considered as viable solutions for the secure platform problem in this paper.

7 Recommendations

Having the technical approaches to address the secure platform problem for remote Internet voting in mind, one may conclude that the combined use of code sheets and test urnes provides a reasonable and practical approach. This is particularly true for environments that already support absentee balloting (e.g., voting by postal mail). It is not necessarily true for environments that do not support absentee balloting. The use of code sheets is particularly well suited for an environment in which the state already provides physical and personalized material (e.g., voting cards) to the voters, using, for example, postal mail. In this case, every mail delivery can also include a personalized code sheet.

There are at least three possibilities to implement code voting. For example, there is the possibility to fully implement code voting using code numbers and verification numbers (i.e., full implementation). There is, however, also the possibility to use either only code numbers (i.e., „code number-only“ implementation) or verification numbers (i.e., „verification number-only“ implementation). Among these possibilities, a „verification number-only“ implementation is particularly interesting, because the voter has to minimally change his or her behavior (i.e., he or she can still enter “YES” or “NO” and only has to validate the verification number sent back from the serber). This advantage, however, may also be a disadvantage, because it is possible and very likely that some voters wont care about the validity of verification numbers sent back from the server.

Code voting makes use of code and verification numbers. The numbers need not be long; their length must only make the probability to correctly guess a number sufficiently small. For example, if the number includes 10 binary digits (bits) the probability to correctly

---

[22] In 1999, the 15 years old Jon Johansen created the DeCSS (De Contents Scramble System) program so that he could view his DVDs on a Linux machine. DeCSS defeats the copyright protection system known as Contents Scramble System (CSS), which the entertainment industry uses to protect films distributed on DVDs. Johansen created and published DeCSS as part of an open source development project to build Linux DVD players called LiViD, or Linux Video.

guess a number is $1/2^{10}$ = 1/1'024 = 0.000975562 ≈ 0.01%. Due to the fact that the numbers can't be verified off-line, this seems to be sufficient. 10 bits can be represented with log 2^{10} = log 1'024 decimal digits which is slightly more than 3 digits. Consequently, 4 decimal digits can be used to encode a code number and some redundancy to detect errors.[23]

There are many possibilities to generate 10-bit numbers. For example, one possibility is to use a keyed one-way hash function and to truncate the results to 10 bits.[24] In the sequel, the term h(K,M) is used to refer to the result of a keyed one-way hash function for the message M (h is a one-way hash function and K is a secret key). This result represents a MAC. In the literature, there are many proposals to compute and verify MACs (e.g., the HMAC construction as specified in [KBC97]).

To implement code voting, it is assumed that there is one or two unrelated and independent cryptographic keys (i.e., K_1 and K_2) for each voting process. The keys must be randomly chosen and kept secret (i.e., only the voting server(s) must have access to the keys). It is further assumed that the string M refers to the concatenation of a reference number for the vote and a reference number for the voter or voting card. In this case, the code number for choice (with choice being „YES“ or „NO“ in the case of a vote and a candidate's name in the case of an election) can be computed as trunc(h(K_1,M|choice)), and the verification number can be computed as trunc(h(K_2|choice)). In either case, | refers to the concatenation and trunc refers to a function that truncates the argument to a specific length. For code numbers (verification numbers), the length is 10 bits (13 bits). For code numbers, 3 bits of redundancy are added to make it possible to detect errors. Because voters are not required to type in verification numbers, the use of redundancy to detect errors is not needed for these numbers.

8 Conclusions and Outlook

In this paper, we argued that there are many possibilities to implement Internet voting, and that remote Internet voting is particularly interesting and challenging. It is challenging, mainly

---

[23] The redundancy scheme is not further addressed in this paper.

[24] Note that one-way hash functions typically provide hash values that are 128 (MD5) or 160 (SHA-1) bits long, and that these values canbe represented in 32 or 40 hexadecimal characters.

because the voter casts his or her ballot at a private site whether he or she (or a third party acting on behalf of the voter) administers and controls the the voting client, platform, and operating environment. This opens the problem that the client platform may be compromised by malicious software to change the voter's ballot before casting it. This problem is referred to as the secure platform problem.

Among the technical approaches to address the secure platform problem for remote Internet voting, the combined use of code sheets and test urnes looks promising. The major advantage (in addition to security) is that the use of code sheets does not need additional hardware or software on the client side. Furthermore, the use of code sheets is similarly applicable to wireless environments, using, for example, mobile phones and WAP devices. As such, the use of code sheets is as technically neutral as possible.

The major open question with regard to the use of code sheets is its usability: will voters accept the new behavior to cast a vote? Instead of simply writing „YES" or „NO" or crossing a corresponding checkbox they will have to write down or type in a number that looks like a 4-digit random number to them. It is an open question whether they understand and will be willing to adapt this behavior.[25] Using, for example, a „verification number-only" implementation in a first step, may provide an easy way to get used to the new voting behavior. The problem with a „verification number-only" implementation, however, is that voters may not check the verification number sent back from the voting server. It will be interested to see the use of code sheets deployed in practice and to learn from the experiences.

9 Acknowledgments

The author thanks the Chancellory of the State of Geneva to give the possibility to work in this interesting and challenging area. He also thanks Michel Warynski, Bernard Taschini, and Jean-Paul Kroepfli for the interesting discussions about possible implementations.

10 Literatur

[BJR01] Shuki Bruck, David Jefferson, and Ronald L. Rivest: A Modular Voting Architecture, Proceedings of the Workshop on Trustworthy Elections (WOTE '01), August 2001.

---

[25] There may also be a legal issue. Entering a number instead of „YES" or „NO" is not possible in all legislations.

[Cal00] California Secretary of State, California Internet Voting Task Force, Final Paper, January 2000.

[Cha01] David Chaum: SureVote: Technical Overview, Proceedings of the Workshop on Trustworthy Elections (WOTE '01), presentation slides, August 2001.

[Hir01] Martin Hirt: Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting, Ph.D. Thesis, ETH Zurich, Reprint as Vol. 3 of ETH Series in Information Security and Cryptography, Hartung-Gorre Verlag, Konstanz, 2001.

[IPI01] Internet Policy Institute: Paper of the National Workshop on Internet Voting: Issues and Research Agenda, March 2001.

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti: HMAC: Keyed-Hashing for Message Authentication, Request for Comments 2104, February 1997.

[Opp02a] Rolf Oppliger: Internet and Intranet Security, 2nd Edition, Artech House, Norwoord, MA, 2002.

[Opp02b] Rolf Oppliger: Security Technologies for the World Wide Web, 2nd Edition, Artech House, Norwoord, MA, to appear in November 2002.

[Riv01] Ronald L. Rivest: Electronic Voting, Proceedings of Financial Cryptography '01, February 2001.

[Rub01] Aviel D. Rubin: Security Considerations for Remote Electronic Voting over the Internet, Proceedings of the 29th Research Conference on Communication, Information and Internet Policy (TPRC2001), October 2001.

[Sal88] Roy G. Saltman: Accuracy, Integrity, and Security in Computerized Vote-Tallying, Institute for Computer Sciences and Technology, NBS Special Publication 500-158, Gaithersburg, MD, August 1988.

[Sch00] Berry Schoenmakers: Fully Auditable Electronic Secret-Ballot Elections, Xootic Magazine, July 2000, Vol. 8, No. 1

[Tho84] Ken Thompson: Reflections on Trusting Trust, Communications of the ACM, Vol. 27, No. 8, August 1984, pp. 761-763