

# Addressing the Secure Platform Problem for Remote Internet Voting in Geneva

Rolf Oppliger, Ph.D.

eSECURITY Technologies Rolf Oppliger  
Beethovenstrasse 10  
CH-3073 Gümligen  
Switzerland

WWW: <http://www.esecurity.ch>  
E-mail: [rolf.oppliger@esecurity.ch](mailto:rolf.oppliger@esecurity.ch)

May 3, 2002

This report was prepared on behalf of the Chancellory of  
the State of Geneva

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Internet Voting</b>	<b>8</b>
<b>3</b>	<b>Security Requirements for Internet Voting</b>	<b>10</b>
<b>4</b>	<b>Further Analysis of the Secure Platform Problem</b>	<b>16</b>
<b>5</b>	<b>Technical Approaches to Address the Secure Platform Problem</b>	<b>19</b>
5.1	“Clean” Operating System and Voting Application . . .	20
5.2	Special Security PC Hardware . . . . .	21
5.3	Closed Secure Devices . . . . .	22
5.4	Secure PC Operating Systems . . . . .	22
5.5	Code Sheets . . . . .	23
5.6	Test Ballots . . . . .	24
5.7	Obscurity and Complexity . . . . .	24
<b>6</b>	<b>Discussion of the Committee Report</b>	<b>25</b>
<b>7</b>	<b>Recommendations</b>	<b>27</b>
<b>8</b>	<b>Possible Implementations</b>	<b>28</b>
8.1	Preliminary Remarks . . . . .	28
8.2	Full Implementation . . . . .	29
8.3	“Code Number-Only” Implementation . . . . .	31
8.4	“Verification Number-Only” Implementation . . . . .	31
8.5	Evolutionary Strategy . . . . .	32
<b>9</b>	<b>Open Questions and Future Work</b>	<b>32</b>
<b>A</b>	<b>Acronyms and Abbreviations</b>	<b>34</b>
<b>B</b>	<b>References</b>	<b>35</b>
<b>C</b>	<b>Relevant Web Sites</b>	<b>37</b>
<b>D</b>	<b>About the Author</b>	<b>38</b>

## Management Summary

In January 2002, the chancellory of the state of Geneva published the final report of a committee that was appointed to study and evaluate the security of a remote Internet voting application [Gen02]. The report came to the conclusion that the application has a reasonable level of security, but that there are at least two related security problems that must be considered with care.

**Server Authenticity Problem:** Internet users normally don't care about the authenticity of servers and don't properly verify public key certificates accordingly. This problem also applies to servers used for remote Internet voting.

**Secure Platform Problem:** The remote Internet voting clients and their platforms are vulnerable to malicious software (e.g., computer viruses, Trojan horses, . . . ) and can be attacked accordingly.

As a possible solution to these problems, the committee report suggests to have the state of Geneva distribute CD-ROMs to voters. The CD-ROMs, in turn, are to include the software and the public key certificates required to authenticate the remote Internet voting server(s) and to remotely vote in a sufficiently secure way. The report further distinguishes between three approaches to configure and distribute the CD-ROMs.<sup>1</sup>

Against this background, the chancellory of the state of Geneva appointed eSECURITY Technologies Rolf Oppliger to study and comment on the suggestions made in the committee report, and to come up with an expert opinion and some recommendations on how to properly address the above-mentioned security problems for remote Internet voting in Geneva. The expert opinion is to take into account and put into perspective the current state-of-the-art in computer security as it relates to the server authenticity problem and—more importantly—to the secure platform problem.

This report is the result of this appointment. It argues that the secure platform problem is known to be hard in the scientific community,

---

<sup>1</sup>In either case, any approach that makes use of CD-ROMs in a sufficiently secure way must require that the CD-ROMs are bootable. Unfortunately, this requirement is not explicitly expressed in the committee report.

and that it is widely believed that is not possible today to implement remote Internet voting in a sufficiently secure way on a large scale. This line of argumentation, however, only makes sense and applies to an environment that does not provide support for absentee balloting. Since the state of Geneva has been supporting absentee balloting in the form of voting by postal mail for a decade, the line of argumentation does not necessarily apply. In fact, one may argue that the security level of any new voting mechanism, including, for example, remote Internet voting, must only be comparable to the security level of voting by postal mail. This level of security seems feasible.

Against this background, the report elaborates on some technical approaches to address the secure platform problem and recommends a combined use of code sheets<sup>2</sup> and test urnes (note that the use of code sheets is in contrast to the committee report that and its suggested use of CD-ROMs that are distributed by the state of Geneva). The basic idea of using code sheets is that constant values (i.e., “YES” and “NO” in the case of a vote, or the candidates’ names in the case of an election) are replaced with variable values that can be guessed correctly by malicious software only with a probability that is arbitrarily small (i.e., negligible). Consequently, it can be assumed that malicious software is not able to modify a vote in some meaningful way. Furthermore, using additional verification numbers that are sent back from the voting server the voter can make sure that he or she is connected to a legitimate server and that his or her vote has actually been received. As such, the use of code sheets is appropriate to address both problems mentioned above (i.e., the server authenticity and secure platform problems). The use of public key certificates for the voting server(s) is still possible and recommended, but it can now be complemented with another server authentication mechanism.

The implementation of code voting appears to be feasible in Geneva, because the state must provide personalized voting cards<sup>3</sup> to the voters using the postal service anyway. These voting cards can then be sent together with the code sheets required to remotely vote. The report overviews, discusses, and puts into perspective three possible implemen-

---

<sup>2</sup>The use of code sheets requires the voter to enter a code number (i.e., a character string) instead of “YES” or “NO” in the case of a vote, or a candidate’s name in the case of an election. In the literature, the use of code sheets for voting is sometimes also referred to as “code voting.” This term is also used in this report.

<sup>3</sup>The voting cards must be personalized because they include a unique secret code that serves as personal identification number (PIN) for voter authentication.

tations of code voting in Geneva and argues in favour of an evolutionary strategy.

There are a few open questions that must be answered before code voting can be deployed and used in the state of Geneva:

- First, it must be verified whether code voting as outlined in the report is legally accepted in Switzerland and the state of Geneva. This must be verified for each possible implementation separately.
- Second, the preliminary notes regarding possible implementations must be expanded into a concept that elaborates on how to properly implement and securely use code voting in the state of Geneva.
- Third, code voting also requires a modified behavior on the voter's side. Instead of writing "YES" or "NO" or simply clicking into a corresponding checkbox, the voter must enter a code number and/or verify another number that is sent back from the server. This is a usability issue and must be treated accordingly. This basically means that field studies have to show whether this modified voter behavior is properly understood by the voters and whether it is accepted in practice.

In either case, the level of assurance for the security of the remote Internet voting application must be increased as much as possible. This requires an open design and an open discussion, as well as peer reviews and audits. It does not mean, however, that all source code must be inspected or all software must be published as open source.



# 1 Introduction

Elections and votes are at the heart of all democracies. In fact, they are important building 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 be able to cast their ballots in some predefined way.

In the literature, the term *electronic voting*, or *e-voting* in short, is used to refer to elections and votes that are supported by electronic means. Independent from the 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,<sup>4</sup> 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.

For the purpose of this report, 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.<sup>5</sup> The election involved several thousands of

---

<sup>4</sup>A list of U.S. patents related to e-voting can be found, for example, at <http://www.safevote.com/patents/>.

<sup>5</sup>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>.

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 only). 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 yet. This fact should be considered with care, because the first state that officially certifies an e-voting system will be at the center of the international media attention.

## 2 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.

- *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.
- 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 voting client, platform, and operating environment.

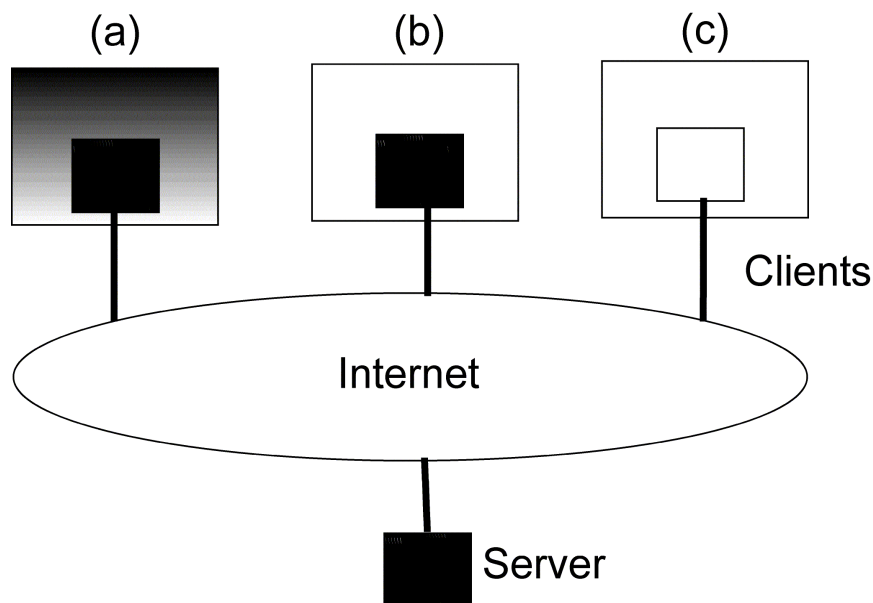
Considering the media attention that has focused on the prospect of using the Internet to vote, it is not surprising that the terms “Internet voting” and “remote Internet voting” are being used synonymously in the popular press. As discussed later, however, it makes a lot of sense to cleanly distinguish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

In some references (e.g., [Cal00]), an additional 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 report and both possibilities are collectively referred to as poll-site Internet voting.

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 these people. 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.<sup>6</sup>



**Figure 1** The three possibilities to implement Internet voting: (a) poll-site Internet voting, (b) kiosk voting, and (c) remote Internet voting. A dark rectangle represents the fact that a client or server and its platform are administered by election officials or that they are operated in a trustworthy environment. Contrary to that, a white rectangle represents the fact that a client and its platform are not administered by election officials or that they are operated in an environment that is not considered to be trustworthy.

The three possibilities to implement Internet voting are illustrated in Figure 1. In either case, a voting client is connected to the Internet (and a voting server) in one way or another. On the client side,

<sup>6</sup>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.

the small rectangle represents the client and its platform, whereas the large rectangle represents the operating environment. A dark rectangle represents the fact that the client and its platform are administered by election officials or that they are operated in a trustworthy environment. Contrary to that, a white rectangle represents the fact that the client and its platform are not administered by election officials or that they are operated in an environment that cannot be considered to be trustworthy. In poll-site Internet voting (a), the client and its platform are administered by election officials and operated in a trustworthy environment at the poll-site (two dark rectangles). In kiosk voting (b), the client and its platform are administered by election officials (dark rectangle), but they are operated in an environment that is not considered to be trustworthy (white rectangle). In remote Internet voting (c), neither are the client and its platform administered by election officials, nor are they operated in a trustworthy environment (two white rectangles). As discussed below, all three possibilities to implement Internet voting have very specific security properties and implications.

### **3 Security Requirements for Internet Voting**

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 show that security (including privacy and reliability) is among the most 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 would set 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, a group of partisans, a 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 lose confidence and trust in the system (even before anything happens).<sup>7</sup>

When talking about security, there are several requirements that must be considered with care. For example, a set of eleven security-related principles have been appended to the mandate of the committee [Gen02]. The following list of security requirements slightly differs from this set of principles (the list is not intended to be complete and comprehensive):

1. Completeness and soundness of the voting protocol;
2. Correctness of the results;
3. Authenticity of the voter (or the voting client acting on behalf of the voter, respectively);
4. Authenticity of the voting server;
5. Secrecy of the ballots (including, for example, anonymity of the voter);
6. Integrity of the ballots (including, for example, protection against malicious software<sup>8</sup>);
7. Non-duplication of the ballots;
8. Availability and reliability of the voting process (including, for example, protection against denial-of-service attacks).

---

<sup>7</sup>Note that confidence and trust are properties that are hard and time-consuming to establish, but they can be lost very rapidly.

<sup>8</sup>*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 quickly that it cannot be stopped. Malicious code is also known as *malware* or *vandalware*. These terms, however, are not used in this report. 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.

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 is 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 also important to note that the security requirements of e-voting are fundamentally different and more difficult to satisfy than the ones of 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.<sup>9</sup> 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.<sup>10</sup>

Most security requirements of Internet voting can be addressed with existing technologies, mechanisms, and services (e.g., [Opp00,Opp02]). For example, the authenticity of the voter can be addressed with secret codes that are distributed on personalized voting cards. Alternative technologies could make use of private keys and 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 as proposed by the designers of the remote Internet voting application of Geneva. It is important, however, 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

---

<sup>9</sup>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.

<sup>10</sup>The fundamental difference between e-commerce and e-voting is also correctly pointed out in [Gen02].

the secrecy and integrity of the ballots before and after they are transmitted over the Internet. This requirement is independent from who actually provides the SSL/TLS implementation (i.e., the browser or a Java applet running in a Java virtual machine (JVM)). Consequently, there are some additional risks for the secrecy of the ballots (i.e., privacy risks) in the home and institutional settings:

- In the home setting, there are some additional privacy risks related to the local use of spyware.<sup>11</sup>
- In the institutional setting, there are some additional privacy risks related to the fact that computer systems and networks are typically managed by remote administrators (rather than the users themselves). The tools these people use for remote administration can also be used to spy on other users' activities.

There are only a few technical approaches that can be used to protect against the (mis)use of spyware and remote administration tools. Fortunately, the use of code sheets as introduced in Section 5.1 is among them.

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 very 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.

---

<sup>11</sup>*Spyware* is software that can be used by one user to spy on the activities of another user (on the same or—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>.

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<sup>12</sup> 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].

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 within creates unique security concerns;
- That currently available client software is far too vulnerable to be used for remote Internet voting;
- That some decades of 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,<sup>13</sup> Bruce Schneier suggested

---

<sup>12</sup>Ronald L. Rivest is the coinventor of the RSA public key cryptosystem.

<sup>13</sup>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>.

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 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,<sup>14</sup> 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 Frequently Asked Questions (FAQ) document<sup>15</sup> of the European *CyberVote*<sup>16</sup> project, 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.”

---

<sup>14</sup>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.

<sup>15</sup>[http://www.eucybervote.org/faq\\_security.html#q35](http://www.eucybervote.org/faq_security.html#q35)

<sup>16</sup><http://www.eucybervote.org/>

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. Since the state of Geneva has been supporting absentee balloting in the form of voting by postal mail for a decade, it does not necessarily apply. In fact, one may argue that the security level of any new voting mechanism, including, for example, remote Internet voting, must only be comparable to the security level of voting by postal mail. This level of security seems feasible.

## 4 Further 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<sup>17</sup> (PC) and a general-purpose operating system, such as Windows, UNIX, 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).<sup>18</sup> Users routinely change the systems' functionalities by adding software modules, such as upgrades, patches, device drivers, dynamic link library (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 even more openly designed and more casually modified through the addition of soft-

---

<sup>17</sup>In this report, the term personal computer is meant to include personal digital assistants (PDAs) with corresponding operating systems.

<sup>18</sup>Note that an open software system is not the same as a system that uses open source software.

ware modules (e.g., plug-ins, Java applets, ActiveX controls, JavaScript scripts, . . . ).<sup>19</sup> 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 reported in the press. For example, the German Chaos Computer Club (CCC) demonstrated an ActiveX control that could initiate and queue up an electronic funds transfer (EFT) 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 possible and very likely that ActiveX controls can be written and deployed that operate more stealthily and are more dangerous accordingly. The same is true for all programming and scripting languages 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<sup>20</sup> 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.<sup>21</sup> 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 necessary to operate a contemporary PC.

Obviously, the same line of argumentation about the untrustwor-

---

<sup>19</sup>Sometimes, this type of software is called "mobile code." This term is not used in this report, 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.

<sup>20</sup>Ken Thompson is one of the developers of the UNIX operating system.

<sup>21</sup>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 include malicious code. Source code inspection does provide, however, a general impression about the style of programming and its security and safety properties.

thinness of code applies for both software that is licensed and software that is published as open source. In fact, it is hard to tell whether open source software is more or less secure and safe than licensed software. The mere fact that software is published as open source does not necessarily mean that many people look at the code, and that these people are qualified to review code. In fact, most open source software developers are more interested in adding new features to an existing software base than to study and thoroughly review it. In either case, the security and safety properties of a software module must be studied on a case-by-case basis, and this study must be independent from its licensing model.

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.<sup>22</sup> 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 software that are assumed to be “clean.”

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 also 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 rogue 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.

---

<sup>22</sup>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.

## 5 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., a 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 remains to 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]:<sup>23</sup>

1. “Clean” operating system and voting application;
2. Special security PC hardware;
3. Closed secure devices;
4. Secure PC operating systems;
5. Code sheets;
6. Test ballots;
7. Obscurity and complexity.

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

---

<sup>23</sup>Not all terms are well chosen. Nevertheless, they are used for consistency reasons.

## 5.1 “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, produced, and distributed by the state or a trustworthy representative thereof.

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

1. The client allows the voter to directly use the Internet to cast his or her vote;
2. 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 voting application client and the server. In the remote Internet voting application of Geneva such a key exists in the form of the secret code.

The major advantage of this approach 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, is 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 many 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 report.

## 5.2 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 entirely 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 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 report.

### 5.3 Closed Secure Devices

Similar to special security PC hardware, it is possible that special, software-closed, Internet-capable devices may be developed and deployed 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 report.

### 5.4 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<sup>24</sup>) or the Extremely Reliable Operating System (EROS<sup>25</sup>). 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 report.

---

<sup>24</sup><http://www.trustedpc.org/home/home.htm>

<sup>25</sup><http://www.eros-os.org>

## 5.5 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 instead 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 this 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. More interestingly, it is possible to only work with verification numbers. The possible implementations of code voting are further addressed in Section 8.

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 in Section 3. 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-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 report. In fact, it is part of the solution that is recommended in Section 7.

## 5.6 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).

Similar to the committee report, the use of test urnes is considered as a viable solution for the secure platform problem in this report. In fact, it is part of the solution recommended in Section 7.

## 5.7 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.<sup>26</sup> Consequently, ob-

---

<sup>26</sup>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),

security and complexity are not considered as viable solutions for the secure platform problem in this report.

## 6 Discussion of the Committee Report

In January 2002, the chancellor of the state of Geneva published the final report of a committee<sup>27</sup> that was appointed to study and evaluate the security of the remote Internet voting application that had been designed and implemented in Geneva [Gen02]. In this report, the committee came to the conclusion that the remote Internet voting application of Geneva has a reasonable level of security, but that there are at least two related security problems that must be considered with care.

**Server Authenticity Problem:** Internet users normally don't care about the authenticity of servers and don't properly verify public key certificates accordingly. This problem also applies to servers used for remote Internet voting.

**Secure Platform Problem:** The remote Internet voting clients and their platforms are vulnerable to malicious software (e.g., computer viruses, Trojan horses, . . . ) and can be attacked accordingly.

As a possible solution to these problems, the committee report suggests to have the state of Geneva distribute CD-ROMs to voters (CD-ROMs have been chosen as media, because they can be made read-only, and because corresponding readers are widely deployed among Internet users). The CD-ROMs, in turn, are to include the software and the public key certificates required to authenticate the remote Internet voting server(s) and to participate in remote Internet voting in a sufficiently secure way. Furthermore, it is suggested to have the client software include a mechanism that automatically authenticates the server(s) in a way that is transparent to the voter.

---

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.

<sup>27</sup>The committee consisted of representatives from the state of Geneva, the European Organization for Nuclear Research (CERN), the hospital of Geneva, and the University of Geneva.

More specifically, the committee report distinguishes between three approaches to configure and distribute CD-ROMs (with increasing levels of security):

1. The CD-ROMs include the most recent versions of standard software (i.e., Web browsers) required to participate in a remote voting process. The software, in turn, is preconfigured to include the public key certificates required to authenticate the voting server(s).
2. The CD-ROMs include client software created specifically for the remote Internet voting application of Geneva.<sup>28</sup>
3. Similar to the second approach. In addition, it is enforced that the client software can be executed only in a minimal and less vulnerable environment.

First of all, it is important to note that all three approaches look similar but are still fundamentally different from the technical approach overviewed and discussed in Section 5.1. In fact, the approach of Section 5.1 requires the PC to boot from a CD-ROM containing a “clean” operating system and voting application. The use of a “clean” operating system is particularly important to protect application software that is expected to run securely and safely on top of it. Simply installing or reinstalling application software does not protect a PC against malicious software (note that the PC may already be compromised and manipulated at the operating system level). Consequently, the mere (re)installation of application software does not provide a viable solution for the secure platform problem as it relates to remote Internet voting. Instead, it must be ensured that the PC boots from an operating system that is assumed to be “clean,” before the application software in question is installed and executed.<sup>29</sup>

Taken these preliminary remarks in account, the approaches enumerated above only make sense if it is enforced that the voter’s PC

---

<sup>28</sup>For reasons that are not obvious, the committee report claims that this approach is similar to the approach followed by the financial industry to provide support for Internet banking. Most Internet banking solutions, however, make use of standard client software.

<sup>29</sup>Another possibility would be to install an operating system from scratch. This possibility is, however, even less comfortable from a voter’s point of view. It is not further addressed in this report.

boots from the CD-ROM. According to a personal conversation, it has been the intention of the committee to make the CD-ROM bootable. This intention, however, is not explicitly expressed in the committee report. Only the third approach can be interpreted in a way that requires bootable CD-ROMs (another interpretation would require Java applets that are executed in a JVM).

Even if one required the CD-ROM to be bootable, the approaches would still have the disadvantages discussed in Section 5.1, and these disadvantages clearly outweigh the advantages. Consequently, the distribution of CD-ROMs cannot be recommended. From a practical point of view, the second and third approaches are even less preferable because they require the state to become a software developer and publisher.

## 7 Recommendations

First of all, it is important to note that the secure platform problem is known to be hard in the scientific community, and that it is widely believed that is not possible today to implement remote Internet voting in a sufficiently secure way on a large scale. This line of argumentation, however, only makes sense and applies in an environment that does not provide support for absentee balloting. Since the state of Geneva has been supporting absentee balloting in the form of voting by mail for a decade, the line of argumentation does not apply. In fact, one may argue that the security level of any new voting mechanism, including, for example, remote Internet voting, must only meet the security level of voting by postal mail. This seems feasible today.

It is recommended to address the secure platform problem with a combined use of code sheets and test urnes (instead of having the state distribute CD-ROMs<sup>30</sup>). The use of code sheets is particularly well suited for the remote Internet voting application of Geneva because personalized voting cards must be provided to the voters anyway. The voting cards are sent using postal mail. Consequently, every mail delivery may also include a code sheet in addition to the voting card. Code sheets have the major advantage that they can be used to address both problems mentioned above (i.e., the server authenticity and secure platform problems). It is further recommended that the code numbers

---

<sup>30</sup>If CD-ROMs are distributed, it is highly recommended to make them bootable (as argued in the previous section).

include some redundancy (i.e, checksums) to detect errors when a voter enters a wrong number.

Some possible implementations of code voting in Geneva are overviewed and briefly discussed in Section 8. The explanations are intended only to clarify the basic ideas. Before a corresponding project is launched, it is recommended to perform a market survey and to look for companies that have developed and are marketing technologies for code voting. For example, a preliminary survey has found a company called SureVote.<sup>31</sup> SureVote was founded by Dr. David Chaum who has a long track record in providing privacy-enhancing and anonymity technologies for the Internet.<sup>32</sup> The technology provided by SureVote is patented in many countries (including, for example, Switzerland) [PCT01].

## 8 Possible Implementations

There are at least three possibilities to implement code voting in Geneva. 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). The possibilities are overviewed and briefly discussed next.

### 8.1 Preliminary Remarks

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 guess a number is  $1/2^{10} = 1/1'024 = 0.000975562 \approx 0.01\%$ . Due to the fact that the numbers can't be verified off-line, this seems to be sufficient. 10 bits can be represented with  $\log 2^{10} = \log 2^{1'024}$  decimal digits which is slightly more than 3 digits. Consequently, 4 decimal digits can be

---

<sup>31</sup><http://www.sure-vote.com>

<sup>32</sup>For example, David Chaum is the inventor of a blind signature scheme that is widely used to implement anonymous digital cash. Prior to his work for SureVote, Chaum founded and led the Dutch company DigiCash to market this technology.

used to encode a code number and some redundancy to detect errors.<sup>33</sup>

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.<sup>34</sup> 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 the reference number for the vote and the reference number for the 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  $trunc(h(K_2, M | choice))$ . 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.

## 8.2 Full Implementation

In a full implementation, each voter receives a personalized voting card and a ballot using postal mail. The voting card consists of the following two parts:<sup>35</sup>

- One part includes the *voting card* that is similar to the one that has been used so far. It includes the following four items:

---

<sup>33</sup>The redundancy scheme is not further addressed in this report.

<sup>34</sup>Note that one-way hash functions typically provide hash values that are 128 (MD5) or 160 (SHA-1) bits long, and that these values can be represented in 32 or 40 hexadecimal characters.

<sup>35</sup>The card can be designed in a way that it can be torn apart before the voting actually takes place.

- A reference number for the vote;
- A reference number for the voting card;
- A secret code;
- A server verification code.

The reference number for the voting card is already defined and consists of 12 decimal digits (the digits are split into three groups of four digits each). Similarly, the secret code already exists and consists of 6 alpha-numerical digits. The secret code is covered and must be scratched off by the voter. In addition to the reference number for the voting card and the secret code, the voting card also comprises a reference number for the vote and a server verification code. The voting date (in some defined format) may be used to reference the vote and another string that consists of 6 alpha-numerical digits may serve as a server verification code. Similar to the reference number for the voting card and the secret code, the server verification code is unique for each voter. It is recommended to use the same character set to encode the server verification code as is used to encode the secret code. Contrary to the secret code, the server verification code need not be covered and must not be scratched off by the voter.

- The other part includes the *code sheet* that is used to actually cast the ballot over the Internet. For each question, the code sheet must include the following four items:
  - A code number to cast a “YES;”
  - A code number to cast a “NO;”
  - A verification number to cast a “YES;”
  - A verification number to cast a “NO.”

As discussed in the preliminary remarks, each code or verification number comprises 4 decimal digits (e.g., 3567).

If a voter decides to cast a vote at poll-site or use postal mail, he or she tears apart the code sheet from the voting card and physically destroys the code sheet.<sup>36</sup> If, however, the voter decides to remotely

---

<sup>36</sup>This step is not critical from a security point of view.

vote over the Internet, he or she has to identify himself or herself to the voting server using his or her reference number for the voting card. Afterwards, he or she must use the code sheet to translate his or her answers (i.e., “YES” or “NO”) into corresponding code numbers, and to verify the verification numbers that are returned by the server accordingly. The server stores the coded answers in a database. If the voter is sure that he or she wants to cast the vote, he or she presses a corresponding button and authenticates himself or herself using the secret code. To make sure that he or she is connected to an authentic voting server, the voter must also verify the server verification code that is returned by the server. This code must match the server verification code printed out on the voting card (note that this check is complementary and must be performed in addition to the server authentication that makes use of public key certificates).

### 8.3 “Code Number-Only” Implementation

If one is not too worried about the server authenticity problem, one can also work with code numbers only. In this case, however, it is still recommended that a voter verifies the public key certificate that is provided by a voting server.

### 8.4 “Verification Number-Only” Implementation

Another possible implementation works with verification numbers only. In this case, the voter must not enter a code number for each question. Instead, he or she enters “YES” or “NO” (or clicks into a corresponding checkbox) and the server sends back the verification number.<sup>37</sup> Again, this number must be verified by the voter.

From a functional point of view, this implementation seems to be equivalent to the full implementation. There are, however, two subtle differences:

1. In a “verification number-only” implementation the voter uses a traditional interface to cast his vote. This basically means that he or she is asked to click into checkboxes. The verification numbers are then sent back to the voter and the voter must manually verify the numbers. Unfortunately, there are no technical means

---

<sup>37</sup>Note at this point that it is very important that the verification numbers for “YES” or “NO” are distinct.

to enforce this manual verification step. If somebody doesn't care about the authenticity of the server, there is no possibility to decide whether something malicious has happened. Note that this is different from the use of code numbers (where the voter must enter a code number and where it is not possible to ignore the new style of voting).

2. The use of code numbers provides secrecy of the ballots even in the presence of spyware and remote administration tools. Obviously, a “verification number-only” implementation does not provide this level of secrecy (because it does not employ code numbers).

These differences should be considered with care before one opts for a “verification number-only” implementation.

## 8.5 Evolutionary Strategy

The possible implementations overviewed above have advantages and disadvantages. From a security point of view, the full implementation is preferred. A “code number-only” implementation does not make a lot of sense, because it does not simplify things considerably. Contrary to that, a “verification number-only” implementation simplifies user behavior and seems to be a good candidate to enter the field of code voting.

In fact, there is an evolutionary strategy that starts with a “verification number-only” implementation and leads to a full implementation. Such a strategy can be recommended for Geneva. In this case, however, test urnes become even more important (to detect if anything goes wrong).

## 9 Open Questions and Future Work

There are a few open questions that must be answered before code voting can be implemented and used in the state of Geneva.

- First, it must be verified whether code voting as outlined in the report is legally accepted in Switzerland and the state of Geneva.<sup>footnote</sup>Note that using code voting, a voter does neither

express “YES” or “NO”, nor does he or she click into a corresponding checkbox. Instead, the voter must enter a randomly-looking code number into a field that is part of a graphical user interface (GUI). This is something entirely new and it must be verified whether it conforms to the legal situation. This must be verified for each possible implementation separately.

- Second, the preliminary notes regarding possible implementations (given in Section 8 must be expanded into a concept that elaborates on how to properly implement and securely use code voting in the state of Geneva.
- Third, code voting also requires a modified behavior on the voter’s side. Instead of writing “YES” or “NO” or simply clicking into a corresponding checkbox, the voter must enter a code number and/or verify another number that is sent back from the server. This is a usability issue and must be treated accordingly. This basically means that field studies have to show whether this modified voter behavior is properly understood by the voters and whether it is accepted in practice.<sup>38</sup>

The first question must be answered before any further effort should be spent on the other two questions.

In either case, the level of assurance for the security of the remote Internet voting application must be increased as much as possible. This requires an open design and an open discussion, as well as peer reviews and audits. It does not mean, however, that all source code must be inspected or all software must be published as open source.

---

<sup>38</sup>Against this background, it is important to note that any new democratic process or procedure (e.g., remote Internet voting) warrants an extremely high level of security, but that the security measures put in place may not be so cumbersome to voters that the new process prevent participation

## A Acronyms and Abbreviations

ACM	Association for Computing Machinery
ATM	Automatic Teller Machine
BIOS	Basic Input Output System
BO2K	Backorifice 2000
CCC	Chaos Computer Club
CD	Compac Disk
CSS	Contents Scramble System
DeCSS	De Contents Scramble System
DLL	Dynamic Link Library
CERN	European Organization for Nuclear Research
DVD	Digital Versatile Disc
EFT	Electronic Funds Transfer
EROS	Extremely Relibale Operating System
FAQ	Frequently Asked Questions
GUI	Graphical User Interface
HMAC	Hashed Message Authentication Code
IDS	Intrusion Detection System
IFIP	International Federation for Information Processing
IT	Information Technology
JVM	Java Virtual Machine
MAC	Message Authentication Code
PC	Personal Computer
PCT	Patent Cooperation Treaty
PDA	Personal Digital Assistant
PIN	Personal Identification Number
ROM	Read Only Memory
RSA	Rivest, Shamir and Adleman
SSL	Secure Sockets Layer
TC	Technical Committee
TCPA	Trusted Computing Platform Alliance
TLS	Transport Layer Security
U.S.	United States
WG	Working Group

## B References

- [BJR01] Shuki Bruck, David Jefferson, and Ronald L. Rivest, “A Modular Voting Architecture,” *Proceedings of the Workshop on Trustworthy Elections (WOTE '01)*, August 2001  
<http://theory.lcs.mit.edu/~rivest/BruckJeffersonRivest-AModularVotingArchitecture-doc.pdf>
- [Cal00] California Secretary of State, California Internet Voting Task Force, Final Report, January 2000  
<http://www.ss.ca.gov/executive/ivote/>
- [Cha01] David Chaum, “SureVote: Technical Overview,” *Proceedings of the Workshop on Trustworthy Elections (WOTE '01)*, presentation slides, August 2001  
<http://www.vote.caltech.edu/wote01/pdfs/surevote.pdf>
- [Gen02] Chancellory of the State of Geneva, Rapport du Comité Sécurité sur l’application de vote par Internet, January 2002  
[http://www.geneve.ch/chancellerie/E-Government/data/rapport\\_securite\\_internet.pdf](http://www.geneve.ch/chancellerie/E-Government/data/rapport_securite_internet.pdf)
- [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, ISBN 3-89649-747-2, 2001
- [IPI01] Internet Policy Institute, Report of the National Workshop on Internet Voting: Issues and Research Agenda, March 2001  
[http://www.internetpolicy.org/research/e\\_voting\\_report.pdf/](http://www.internetpolicy.org/research/e_voting_report.pdf/)
- [KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti, “HMAC: Keyed-Hashing for Message Authentication,” Request for Comments 2104, February 1997  
<http://sunsite.cnlab-switch.ch/ftp/doc/standard/rfc/21xx/2104>
- [Opp00] Rolf Oppliger, *Security Technologies for the World Wide Web*,<sup>39</sup> Artech House, Norwood, MA, ISBN 1-58053-045-1, 2000

---

<sup>39</sup>A second edition of this book will be published in 2003

<http://www.esecurity.ch/Books/wwwsec.html>

[Opp02] Rolf Oppliger, *Internet and Intranet Security, Second Edition*, Artech House, Norwood, MA, ISBN 1-58053-166-0, 2002  
<http://www.esecurity.ch/Books/iis2e.html>

[PCT01] David Chaum, *Physical and Digital Secret Ballot System*, International Patent published under the Patent Cooperation Treaty (PCT), WO 01/55940 A1, August 2, 2001

[Riv01] Ronald L. Rivest, "Electronic Voting," *Proceedings of Financial Cryptography '01*, February 2001  
<http://theory.lcs.mit.edu/~rivest/Rivest-ElectronicVoting.pdf>

[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  
<http://avirubin.com/e-voting.security.html>

[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  
<http://www.itl.nist.gov/lab/specpubs/500-158.htm>

[Sch00] Berry Schoenmakers, "Fully Auditable Electronic Secret-Ballot Elections," *Xootic Magazine*, July 2000, Vol. 8, No. 1  
<http://www.win.tue.nl/xootic/magazine/jul-2000/schoenmakers.pdf>

[Tho84] Ken Thompson, "Reflections on Trusting Trust," *Communications of the ACM*, Vol. 27, No. 8, August 1984, pp. 761-763  
<http://www.acm.org/classics/sep95/>

## C Relevant Web Sites

- CalTech-MIT Voting Technology Project  
<http://www.vote.caltech.edu>
- Election.com  
<http://election.com>
- ETH Zürich E-Voting Project  
<http://www.crypto.ethz.ch/research/sdc/vote/>
- European CyberVote Project  
<http://www.eucybervote.org>
- MIT Electronic Voting Bibliography  
<http://theory.lcs.mit.edu/~cis/voting/greenstadt-voting-bibliography.html>
- Rebecca Mercuri's Web site on Electronic Voting  
<http://www.notablessoftware.com/evote.html>
- SureVote  
<http://www.sure-vote.com>
- University of Osnabrück, Research Group "Internetwahlen"  
<http://www.internetwahlen.de>
- VoteHere  
<http://votehere.com>
- Workshop on Trustworthy Elections 2001 (WOTE '01)  
<http://www.vote.caltech.edu/wote01/>

## D About the Author

Rolf Oppliger received his M.Sc. (1991) and Ph.D. (1993) degrees in computer science from the University of Berne, and the *venia legendi* (1999) from the University of Zürich. He is a lecturer at the University of Zürich and occasionally teaches at other universities and polytechnics in Switzerland and Germany. In 1999, he founded eSECURITY Technologies Rolf Oppliger (<http://www.esecurity.ch>) to provide scientific and state-of-the-art consulting, education, and engineering services related to information technology (IT) security. In addition, he works for the Swiss Federal Strategy Unit for Information Technology and is the current editor for the Computer Security Series published by Artech House. He has published nine books<sup>40</sup> and numerous scientific papers, articles, and reports, mainly on security-related topics. He is a member of the ACM, the IEEE Computer Society, and the IFIP TC 11 WG 4 on network security.

---

<sup>40</sup>Two books are referenced in this report.