Improving Remote Voting Security with CodeVoting

Rui Joaquim, Carlos Ribeiro, and Paulo Ferreira

ISEL - Technical University of Lisbon - INESC-ID
rjoaquim@cc.isel.ipl.pt,
carlos.ribeiro@tagus.ist.utl.pt,
paulo.ferreira@inesc-id.pt

Abstract. One of the major problems that prevents the spread of elections with the possibility of remote voting over electronic networks, also called Internet Voting, is the use of unreliable client platforms, such as the voter's computer and the Internet infrastructure connecting it to the election server. A computer connected to the Internet is exposed to viruses, worms, Trojans, spyware, malware and other threats that can compromise the election's integrity. For instance, it is possible to write a virus that changes the voter's vote to a predetermined vote on election's day. Another possible attack is the creation of a fake election web site where the voter uses a malicious vote program on the web site that manipulates the voter's vote (phishing/pharming attack). Such attacks may not disturb the election protocol, therefore can remain undetected in the eyes of the election auditors.

We propose the use of CodeVoting to overcome insecurity of the client platform. CodeVoting consists in creating a secure communication channel to communicate the voter's vote between the voter and a trusted component attached to the voter's computer. Consequently, no one controlling the voter's computer can change the his/her's vote. The trusted component can then process the vote according to a cryptographic voting protocol to enable cryptographic verification at the server's side.

Keywords: Remote voting, Internet voting, vote manipulation, uncontrolled voting platform, insecure voting platform.

1 Introduction

Remote electronic voting over electronic networks, also called Internet voting, is very appealing because it offers the possibility of voting from anywhere with an Internet connection, on election day, avoiding the need to vote in advance. Therefore, remote Internet voting offers a convenient way to vote for users away from home on vacations, due to work, or for any other expected or unexpected reason. However, surprisingly or not, there are not many Internet voting systems in use today. One of the main reasons for this situation is that a computer connected to the Internet does not offer a secure voting environment.

D. Chaum et al. (Eds.): Towards Trustworthy Elections, LNCS 6000, pp. 310–329, 2010.
© IAVOSS/Springer-Verlag Berlin Heidelberg 2010

A computer connected to the Internet is exposed to several threats, such as viruses, worms and Trojans, among others. These threats take advantage of vulnerabilities in software to perform malicious actions, including taking control of the computer. The number of new vulnerabilities reported is too high to be ignored, 8064 in 2006 and 5568 in the first three quarters of 2007 [1,2]. It is easy to imagine a virus exploring one of those vulnerabilities to steal or change the voter's vote.

It is also possible, in some situations, to steal the voter's vote without attacking the voter's computer. An attack to the Internet support infrastructure could send the voter to a fake election web site. Once in the fake election web site the attacker could use the voter's authentication information and steal the voter's vote. Such type of attack is called pharming [3]. An example of an attack technique against home routers can be found in [4]. Reports on recent pharming attacks against banks in the US, Europe and Asia-Pacific can be found in [5,6].

Another important issue concerning remote voting in general is that there are no guarantees that the voter will vote alone. Therefore, the voter may be subject to coercion. A special case of coercion within family members that live in the same house, named family voting, has been observed in several countries [7] and is a problem to consider whenever remote voting is allowed, e.g. postal voting, Internet voting. A problem that is somehow similar to coercion is vote selling. Since the voters cast votes in an uncontrolled environment they can give their ballot to anyone, or vote in the presence of anyone, therefore they may be tempted to sell their vote. A possible protection against vote selling and coercion is to allow the voter to update their vote, i.e. cast several votes [8].

The main difference between traditional remote voting, e.g. postal voting, and Internet voting is that Internet voting attacks are able to target a large number of voters with a fraction of the budget. Our opinion is that an attack to steal/change the voter's vote by attacking the voter's computer or the Internet infrastructure poses a potentially higher risk to the election's integrity than an online vote buying or coercion attack. We base our opinion on the following four reasons:

- First, large scale vote buying/coercion, involving possibly thousands of voters, is quite unlikely to pass undetected. Additionally, vote buying/coercion can be discouraged by allowing vote updates.
- Second, with all the security flaws in operating systems and applications, it is easy to write a virus that would be active on election day to change the voter's vote.
- Third, we believe that writing a virus and disseminating it would be cheaper and more difficult to trace back to the authors than a vote buying/coercion attempt of a thousand voters, therefore, more appealing to an attacker.
- Fourth, punishing the attackers would be very difficult, if not impossible, because the attack could be carried out from anywhere in the world.

Therefore, we can say that the insecure voting platform [9,10,11,12,13], offered by a computer connected to the Internet, and the possibility of an unpunished attack are the main issues that prevent the spread of remote Internet voting.

We propose the use of CodeVoting, a technique to communicate the voter's vote between the voter and a trusted component attached to the voter's computer. Consequently, no one controlling the voter's computer can change the vote. CodeVoting offers a manually verifiable proof of correct vote fulfillment and submission, based on the assumption about the existence of a trusted and secure component attached to the voter's computer. In this paper we present the CodeVoting technique [14,15] and an enhancement to support large candidate lists and multiple elections.

1.1 Why Cryptographic Voting Protocols Are Not Enough

In the last 30 years many cryptographic voting protocols were proposed to secure electronic voting. Several cryptographic techniques were employed such as blind signatures [16,17,18,19], mix-nets [20,21,22,23] and homomorphic ciphers [24,25,26,27]. However, the main concern of these cryptographic voting protocols is the protection against vote manipulation at server side, assuming a trusted client to establish a secure communication channel to the election server and to perform the cryptographic steps and verification of the voting protocol.

It is clear that these cryptographic voting protocols do not work well without the assumption of a trusted voting client. The CodeVoting's goal is to provide the means to secure the use of a cryptographic voting protocol from a generic computer connected to the Internet.

1.2 CodeVoting Overview

CodeVoting does not replace traditional voting protocols. It works by creating a secure channel between the user and a trusted component that runs one of the traditional voting protocols with the voting server (Fig. 1). Therefore, CodeVoting can be considered a kind of user interface for the voting system.

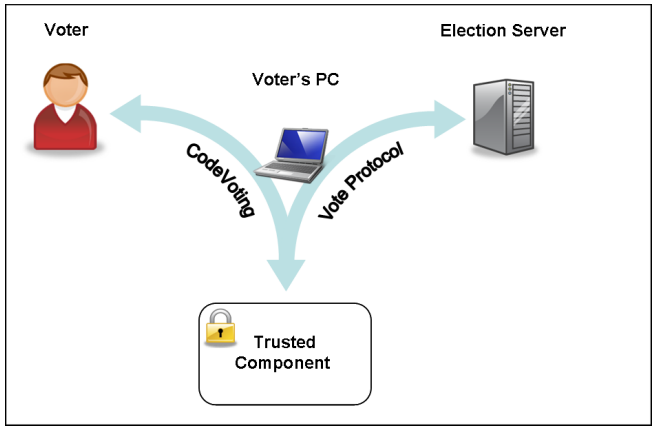

Fig. 1. CodeVoting overview

We propose the use of a tamper resistant device, such as a smart card, to be the trusted component of the voting system at the client's side. Since a smart card provides a much more secure execution platform than an off-the-shelf PC, using CodeVoting to create a secure communication channel to the smart card, through the voter's PC, will prevent automatic vote manipulation by malicious software installed in the voter's PC. Since we propose the use of a cheap tamper resistant device, such as a smart card, that does not have network nor I/O capabilities, the voter's PC network and I/O capabilities will still be used to interact with the voter and the server's side of the voting system.

The use of secure devices, e.g. smart cards, in electronic voting is not a new concept [28,29]. Secure devices are typically used as a way to provide secure voter's authentication and for the generation and secure storage of secret values for a voting protocol. However, it is always necessary to use the voter's computer to show the ballot and collect the answer, and this is usually assumed to be performed by a trusted vote client application. Our goal is to show how one can build a simple, secure and private communication channel between the voters and their trusted devices, without the need to trust in the voters' computers. Secure communication channels are easy to achieve between machines, e.g. by sharing a secret key. However, making a secure and private communication channel between a machine and a human is not so straightforward. The challenge is to keep the complexity of the communication channel as small as possible in order for the human to be able to deal with it.

1.3 Vote Manipulation Attacks

A vote manipulation attack can be a modification of the voter's vote. It can be performed in two ways: i) changing a vote to a predetermined candidate, or ii) changing a vote to a random candidate. While the first attack is more powerful, the second may be easier to prepare in advance. In other words, to change a vote to a predetermined candidate one must have the knowledge of which candidate one wants to change the vote to, while to change the vote to a random candidate there is no need to know the candidates in advance.

If one wants to increase the number of votes for a candidate A it is preferable to perform an attack that will directly change the votes to votes for candidate A. On the other hand, if one wants to decrease the number of votes for a candidate B, it suffices to perform a random vote modification attack in an area known to be more favorable to candidate B.

The other kind of vote manipulation attack is to fake a successful vote delivery. In many voting systems this can be done by just presenting the message "Your vote was successfully delivered. Thank you for voting.". This attack allows an attacker to reduce the votes on a candidate just by targeting an area with great affinity for that candidate.

In the next section we describe the currently proposed approaches to minimize the weaknesses at the client's side. In section 3 we present the CodeVoting technique, a solution that prevents vote manipulation at the client's side by using a mix of "special security PC hardware" and "code sheet" approaches. We evaluate

the CodeVoting proposal's resistance to vote manipulation attacks in section 4. In section 5 we present the Matrix CodeVoting, a CodeVoting enhancement to support large candidate lists and multiple elections. Finally, in section 6, we present the conclusions and future work.

2 Related Work

Cryptographic voting protocols can prevent vote manipulation at server side. However, that is only relevant if the votes cannot be easily manipulated at the uncontrolled client side of a remote voting system. Here we present an overview of the approaches proposed to mitigate the problem of the unreliable vote client platform [9,30].

2.1 Clean Operating System and Voting Application

This approach assumes the existence of a CD-ROM with a "clean" and certified operating system and voting application. The voter should boot her computer from the CD-ROM to have access to the vote application. One of the major problems with this approach is how to design such CD-ROM so that it would allow the voters to boot from any computer in use. Another problem is how to provide Internet access. Voters have different types of Internet connections at home, such as modem, ADSL, cable. Would the voters have to manually configure their connection parameters?

The immediate consequence of such variety of configurations is that a large amount of software, besides the operating system and voting application, has to be on the CD-ROM, and consequently also, has to be verified. Therefore, some questions remain: can we claim that a CD-ROM is clean? And to which extent? Is it clean enough to provide a secure voting platform?

With all these questions/problems pending we do not believe that this approach could be successfully used to enable secure voting from any computer with an Internet connection.

2.2 Special Secure Hardware for a PC

This approach assumes a dedicated software-closed security device, with secure I/O, attached to the voter's computer, e.g. through a USB port. Its purpose is to display the ballot to the user, accept the voter's choices as input, and perform cryptographic operations. In effect, the voting protocol is executed by the secure device. Since the device is software-closed, meaning its software cannot be changed, it is not subject to infection with a malicious code. The main disadvantage of this approach is the cost of such dedicated hardware. Moreover, the manufacturer and the distribution process of the devices must be fully trusted.

Zúquete et al. [31] implemented a system based on this concept. They use a secure smart card terminal with I/O capabilities to display the ballot and read the voter's answer. In addition, they use a smart card to provide public key authentication. The main disadvantage of the system, besides the cost, is the reduced display capacity of the terminal, which is only 4 lines of 20 characters.

2.3 Closed Secure Devices

The use of closed secure devices was one of the proposals made by the California Internet Voting Task Force in 2000 [9]. The proposal was the use of special software-closed and Internet-capable devices, such as network computers (NCs) or hand-held, wireless descendants of those days (early 2000s) cell phones and electronic organizers.

However, modern cell phones and electronic organizers that have Internet access usually allow the user to install arbitrary applications too. This facility makes the systems open to malicious applications that take advantage of the vulnerabilities of the operating system of the device.

Moreover, the use of closed secure devices just to access a web site on the Internet, through which the voter votes, is vulnerable to attacks to the Internet infrastructure, such as pharming attacks.

2.4 Secure PC Operating Systems

This approach suggests the use of secure PC operating systems that may be composed of digitally-signed modules, allowing secure applications to exclude, as untrusted, modules of dubious origins (i.e. potentially malicious programs).

Trusted computing is the name given to the technology that is being developed today to offer such secure platform support [32]. Trusted computing is a technology that allows the remote attestation of machines and programs running on them. With remote attestation it is possible to certify that a voter is using a correct voting program. Trusted computing also provides ways to secure I/O operations between the program and the physical I/O devices, therefore creating a secure environment for an application to run. The attestation process is based on measures performed on the software by a hardware module called trusted platform module (TPM).

The client of a remote voting application needs to interact with the voter (I/O device drivers), needs to establish a connection with a voting server (network protocol stack + network adapter driver) and, last but not least, it needs an environment to run on, i.e. a working operating system. The attestation of the core of the operating system, the device drivers and the voting application can be cumbersome. Moreover, there are also problems concerning the maturity of the currently deployed technology [33] and concerning the revocation of cracked machines [34]. We believe that, for the time being, the application of trusted computing to remote voting as the only guarantee of the correct application behavior is not a valid alternative. Nevertheless, there are proposals to use trusted computing technology to solve the uncontrolled platform problem in remote voting [35].

2.5 Code Sheets

This approach consists in secretly sending, e.g. by mail, code sheets to voters that map their choices to entry codes on their ballot. While voting, the voter uses

the code sheet to know what to type in order to vote for a particular candidate. In effect, the voter does the vote encryption and, since no malicious software on the PC has access to the code sheet, it is not able to change a voter's intentions.

The first code sheet implementation we are aware of was proposed in 2001 by Chaum [36], the SureVote system. SureVote allows the voter to cast a vote using secret vote and reply codes. SureVote generates secret vote and reply codes for each candidate and for each voter. The codes are delivered to the voters prior to the election day. On election day the voter sends the vote code of her/his favorite candidate through the voting channel, e.g. Internet. At server side, the reply code is computed by a set of trustees and sent to the voter that confirms it - in this way it is verified that there was no vote modification. After the election day the trustees compute the real votes from the vote codes and publish the results. However, if there is at least one corrupted trustee, SureVote does not guarantee that in the counting phase the vote code is translated to the right candidate.

A code sheet system was used in the UK on some pilots of Internet, SMS and telephone voting [37,38]. A similar system was also proposed by Helbach and Schwenk [39] in which they suggest the use of a three-way-handshake voting protocol. They use a third code to confirm the vote.

The main drawback of this approach is the difficulty to guarantee that the codes are secretly generated and anonymously delivered to the voters. Another drawback is that there is no guarantee that the code vote is translated to the right candidate at server side, i.e. the reply code only confirms that the vote has reached an entity that knows the right reply code.

2.6 Test Ballots

This approach requires the use of special test ballots to be sent from clients and checked by software at the election authorities' office. The number, location, timing, and contents of the test ballots should be known by the county, but they should be otherwise indistinguishable from real ballots, so that any malicious code that destroys or changes real ballots will affect the test ballots as well. The analysis of the test ballots will enable any malicious code attacks to be detected, the locations of infected machines to be determined, the approximate time of the attack to be estimated, and the total number of votes affected to be bounded. Note that this technique does not prevent malicious code attacks; it only detects them after their occurrence. Hence it must be combined with one of the previously presented techniques.

Of course, this technique only works if the attack is to be performed after the vote is produced by the vote client software. The attack will not be noticed if it just modifies the voter's option before passing it to the method that processes the vote and delivers it to the vote server.

Still, this approach can be used as a kind of intrusion detection system that can detect any systematic cause of lost ballots, not just malicious code attacks, and provide a quantitative measure of the size of any problem it detects.

2.7 Obscurity/Complexity

This approach, while not sufficient to guarantee the security of the system, raises the cost for potential attackers. Digital ballot formats and voting software may be kept secret prior to the election and possibly randomly changed during the election, or made complex in other ways. In order to successfully carry out an attack and escape detection, malicious software authors must have a great deal of information about the internal format of the ballot and voting software. If these details are not available in advance, and/or if that information is complex, the potential authors of attack software may not have enough time to develop and distribute it during the election window.

On the other hand, it is difficult to establish a lower bound to the time needed to write malicious software. Additionally, the system is still vulnerable to pharming attacks that collect voters' authentication data and use them later in the real voting client software.

2.8 External Channel Verification

This approach consists in having a secondary communication channel to the election server that would allow the confirmation of the correct vote delivery. Kutyłowski and Zagórski [40] proposed a voting protocol where a voter uses a secondary channel first to "decrypt" the ballot and choose the candidate, and then to verify with a probability of $\frac{1}{2}$ that the vote was correctly submitted to the election server. The main disadvantage of this protocol is the complexity of the verification tasks given to the voter that must deal directly with the encrypted ballots.

Skagestein et al. [41] proposed the verification of the clear casted vote. In their approach, a voter who wants to verify her/his vote can just use another PC and ask to see the casted vote. This second PC asks the voting server for the voter's vote, opens it with the secret encryption key used to encrypt the vote that should be stored in a secure medium at the time of vote casting, and displays the vote to the voter. To minimize the danger of vote selling and coercion, the authors proposed that the cast of several votes should be allowed; therefore, the vote buyer or coercer would not know if the verified vote was the final one. The main disadvantage of this protocol is that it does not prevent vote manipulation at server side. Additionally, anyone with access to the encrypted ballots, considered to be part of the final tally, can use them as a proof of vote since they can be decrypted using the secret encryption keys kept by the voters.

The main disadvantage of this approach is that it requires the voter to have access to two independent communication channels. Additionally, a verification step sometime after the vote casting procedure is not convenient for voters.

3 CodeVoting

We propose CodeVoting, a solution/system to prevent vote manipulation at client side while allowing the use of cryptographic voting protocols to protect

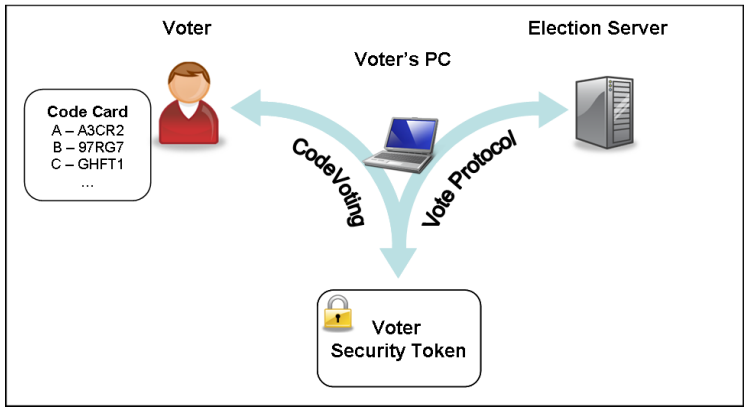

Fig. 2. CodeVoting components' overview

the election's integrity at server side. CodeVoting is a mix of the "special security PC hardware" and "code sheet" approaches, cf. sections 2.2 and 2.5. Figure 2 presents an overview of the CodeVoting components.

CodeVoting proposes the use of a tamper resistant device without any human I/O capabilities, the Voter Security Token (VST), to perform a cryptographic voting protocol at client side and securely authenticate the voter, e.g. by means of digital signatures. The voter communicates her/his choice to the VST using a code sheet approach based on codes printed on a paper card, the CodeCard. The CodeCard is generated directly by the VST, which prevents vote manipulation by a malicious computer. On the other hand, as explained later, the VST also provides a proof of the correct conclusion of the voting protocol.

Briefly, our solution consists in the following steps: i) the voter expresses her/his vote as a secret code, ii) the secret code is translated into the corresponding candidate identifier (ID) (clear vote), iii) the clear vote is used in a cryptographic voting protocol, and iv) once a cryptographic proof of the correct vote delivery is received, a successful vote delivery code is released to the voter.

3.1 Voter Security Token

The VST is the component in charge of the vote code translation to the clear vote as printed in the CodeCard (Sec. 3.2). After the vote code translation it is possible to use it in any voting protocol. The voting protocol runs inside the VST to prevent any vote manipulation at client side. It is possible to use a cryptographic voting protocol to prevent vote manipulation at server side. Usually, cryptographic voting protocols require the use of digital signatures to authenticate the voters. We suggest the use of the VST to enable such authentication mechanism. The VST should be protected by a PIN to prevent unauthorized access.

We propose to distribute the VSTs to the voters in a preliminary registration phase. This procedure is only required once, i.e. the VST will be reused in subsequent elections. To provide a secure voter's authentication mechanism, by

means of a digital signature, a public key infrastructure (PKI) should be in place before the registration process. The PKI can be set up just for election purposes or can be more widely used in a national e-Government project. This last approach can be useful to prevent, at some level, vote buying and coercion, because if the voter gives her/his VST to a vote buyer/coercer it is not just a vote that the s/he gives away, it is also all the e-Government rights of the voter.

3.2 CodeCard

The CodeCard is nothing more than a paper card that associates each candidate ID to a vote code printed on it. There should be one CodeCard per VST so that every voter votes with different codes. The voter should be the only one with access to the codes printed on her/his CodeCard. Consequently, we have a problem to solve: how to create the CodeCard, associate it with the VST and give it to the voter without leaking the codes.

We propose to generate each voter's CodeCard within the VST. This is a viable option because the CodeCard becomes automatically associated with the corresponding VST and no other entity besides the VST has access to the codes on the CodeCard. However, we still have the problem of how to secretly print the CodeCard, i.e. how to give it to the voter without leaking the codes. We believe the best idea is to have a certified CodeCard printing machine, the CodeCard generator interface (CCGI), available at the local authorities' offices. Since the codes are generated inside the VST, the CCGI would be very simple. It would consist of only an interface to the VST, e.g. a smart card reader, in the case of using smart cards, a keypad (for inserting a PIN to unlock the VST) and a small printer. We believe that such simple hardware could be easily certified and sealed to ensure the secrecy of the codes printed. With a certified CCGI in all local authorities' offices a voter can go to any local authority's office and generate a new CodeCard for her/his VST. For privacy reasons the CCGI should be inside a private booth, similar to the ones used for traditional paper-based voting.

3.3 CodeVoting Details

CodeVoting can be seen as a rearrangement of the ideas presented by Chaum [36]. However, the idea of CodeVoting is to only use the codes as a user interface and not as the entire voting protocol. The secret codes are the base for the secure communication channel between the voter and her/his VST. The voter uses secret codes to choose her/his favorite candidate. Each VST has a set of secret codes associated with it that are printed on a CodeCard.

For the voter, the voting process is quite simple. The voter just uses a CodeCard to translate the candidate ID into a vote code. For instance, a voter, with the ballot and CodeCard of Fig. 3, who wishes to vote for a candidate D only has to enter WL764 as the vote code.

Every voter will have a different CodeCard. Therefore, different vote codes exist for the same candidate. Each CodeCard is associated with a VST, which is responsible for the translation of the vote code to the candidate ID. Only the

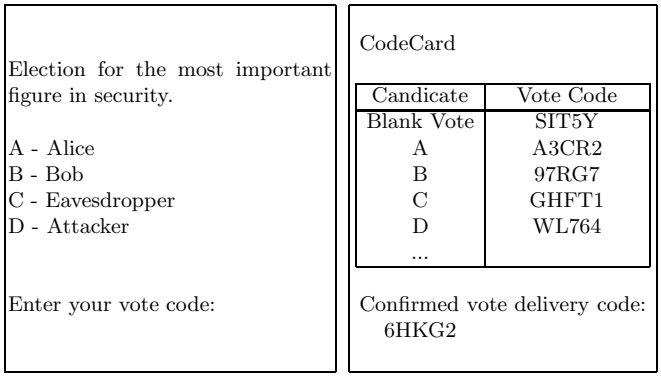

Fig. 3. Example of a ballot (on the left) and a CodeCard (on the right)

voter and the VST should know the codes written on the CodeCard. Therefore, CodeVoting is able to prevent a malicious voting application from changing the voter's vote. Note that there should also be a specific code for a blank or spoiled vote to prevent the malicious voting application from easily casting such a vote.

After translating the vote code to the candidate ID, any voting protocol can be used to cast the vote, e.g. a cryptographic voting protocol that protects the election's integrity at server side. When the VST receives a confirmation of a successful vote delivery from the election server, it releases the confirmed vote delivery code, assuring the voter that her vote was successfully delivered.

Based on the description of CodeVoting the reader can easily understand that CodeVoting is a type of user interface plugin to a voting system that protects the voter's choice from manipulation.

Note that we do not use different reply codes for each candidate. The reason for this is simple, as explained in section 2.5, the code sheet approach offers no guarantees that the vote code is translated to the right candidate at server side, i.e. the reply code only confirms that the vote has reached an entity that knows the right reply code. Therefore, the only advantage of using a different reply code for each candidate is that it makes it more difficult for an attacker to change the vote to a random candidate without being detected by the voter, i.e. the attacker needs to get the correct vote and reply codes.

However, to avoid vote stealing, the length of the vote codes must be defined to prevent an attacker from guessing a valid vote code. Therefore, and from a theoretical point of view, the use of a single reply code is enough to detect an attacker trying to forge a successful vote delivery. Additionally, the use of only one reply code reduces approximately by $\frac{1}{3}$ the amount of information to be printed on the CodeCard.

It is also important to note that CodeVoting is vulnerable to malicious applications that change the correspondence between the candidates and the candidates' IDs. This vulnerability is due to the lack of secure output on the VST. However, there are measures to prevent ballot modification, such as i) publicly exposing the ballot some time before the election, ii) forcing the sorting of the candidates

on the ballot, and the corresponding candidate IDs, in a verifiable way, e.g. alphabetical sorting, and iii) using an image that is hard to forge/modify as a ballot.

4 Evaluation

We argue that CodeVoting protects against vote manipulations at the voter's PC under the following assumptions: i) the VST performs the protocols correctly and cannot be manipulated by the voter's computer, ii) the CodeCard is generated in a secure and controlled environment by the VST (the voter is the only person there), iii) the voter keeps her/his CodeCard secret, and iv) the correspondence between the candidate and its ID cannot be changed.

Under these assumptions, changing a vote to a predetermined candidate is virtually impossible because the corresponding vote code is not known by the attacker, and the *Probability of Guessing the Correct Vote Code* $Pgcvc = \frac{1}{36^5}$, with 5 alphanumeric symbols, i.e. less than 1 in 60 millions. If we use both capital and non-capital letters, we have $Pgcvc = \frac{1}{62^5}$, i.e. less than 1 in 900 millions.

A random change attack has almost the same probability of success as the previous attack. If we have n candidates, the *Probability of Guessing a Random valid Vote Code* is $Pgrvc = n.Pgcvc$.

However, to prevent an easy denial of service attack the VST should not automatically block when the voter inserts invalid vote codes. Therefore, this limitation allows an attacker to perform a brute force attack to get a random valid vote code. Such attack can be minimized through simple measures, such as delaying the vote code verification function, e.g. with a delay of three seconds the attacker is limited to only 1200 tries in a one hour attack. Another possibility for reducing the chances of success of a brute force attack is increasing the domain of the vote codes, either by increasing the code length or by using more symbols, e.g. capital and non-capital letters. For instance, by using 62 symbols, the probability of a successful one hour brute force attack is 1 in 763444 for 5-digit codes, and 1 in more than 47 million for 6-digit codes.

The attacker can also try to manipulate the voter's vote by fooling the voter into believing that s/he has cast a vote, while in reality no vote was cast. To prevent such an attack we propose the use of a code to confirm the vote delivery. The VST only releases this code after getting a confirmation that the vote was successfully delivered, e.g. it could be a message signed by the election server. Therefore, if we use a confirmed vote delivery code with the same length as the vote codes, this attack has the same probability of success as the attack of changing a vote to a predetermined candidate. The confirmation received by the VST can be stored inside it to provide a proof of vote delivery, therefore allowing the voter to protest if her/his vote is not considered for the final tally.

Another important aspect is CodeVoting in the face of vote buying/coercion problems. CodeVoting allows the voter to produce a receipt of the vote by giving away the CodeCard to an attacker prior to the election day. On election day, the attacker can demand the voter to vote using a computer controlled by him/her,

e.g. by using a web site that is controlled by him/her or a special program also developed by the him/her. In this case, the attacker will have a vote receipt that proves the voter's vote, therefore enabling vote buying and coercion. We think the best way to prevent such an attack is by using a voting protocol that allows the voter to cast several votes, i.e. update her/his vote. We believe that the possibility of a vote update in a machine which is not controlled by the attacker would discourage vote buying and coercion attacks [8].

CodeVoting offers protection against vote manipulation and allows the detection of malicious interference in the voting process. However, CodeVoting cannot force a malicious computer to behave properly. Therefore, if any malicious interference in the voting process is detected, the voter should go to another computer and vote again.

4.1 Is the VST Trustworthy?

CodeVoting relies on the correct behavior of the VST. Therefore one can ask: CodeVoting is designed to protect the voter from the insecure voter's PC, but what guarantees are given that the VST does in fact do what it is supposed to do? One way verify it is by testing the VST. Besides testing the VSTs in the production phase, we believe that it would be good to have additional random testing by an independent certification authority.

Additionally, we can also make voters part of the certification process. It could be possible for a voter to verify her/his VST by running a fake election with instant results sometime before the real election.

Nevertheless, one can point out that the application running inside the VST is somehow able to detect that it is being subject to a test, and therefore will act properly in the tests but will still change the voter's vote on the real election day. This scenario can be prevented if one is sure of the software running inside the VST.

Fortunately, today there is secure tamper resistant hardware, such as smart cards, that supports signed applications. Therefore, it is possible to use publicly available and certified source code software. Of course, we can also have certified applications running on a PC. However, since it is possible for an attacker to take control of the voter's PC, a signed application does not guarantee the correct behavior. On the other hand, it is not possible to take control over secure tamper resistant hardware, or at least it should be very difficult even with specialized tools and impossible with just a common computer such as the ones at our homes. Therefore, a signed application running on secure tamper resistant hardware can guarantee the correct behavior.

5 Matrix CodeVoting

The presented CodeVoting technique has some limitations. First, it does not support large candidate lists, so its application is limited. Indeed, the CodeCard must have an entry for each possible candidate. If we consider elections with

a large number of candidates, let's say above thirty, the size of the CodeCard could start to become too large or usability problems may arise, e.g. due to small fonts used.

Another issue is the CodeCard reuse. If a voter uses the same CodeCard in more than one election, it is possible for an attacker to replace the voter's vote by a random one. To be able to do this, an attacker must collect the codes used during the first election. Additionally, the attacker must also be in control of the PCs used by the voter to vote in each election. If the voter uses different PCs, it will be much harder to perform the attack. We can also make this attack harder by executing the whole voting protocol inside the VST, without leaking the voter's identification to the voter's PC. Therefore, the different PCs only have the unlocking PIN to identify the voter.

Of course, the voter can always protect himself/herself against this type of attack by getting a new CodeCard for his/her VST between elections. However, an issue still remains: simultaneous elections. The simultaneous elections' issue is a particular case of the CodeCard reuse, in which the voter cannot go (or is not convenient for him/her to go) to the local authorities and get a new CodeCard. One solution that may work in some particular cases is the use of sequential candidate IDs throughout all the simultaneous elections, e.g. instead of using candidate IDs A and B for elections 1 and 2, use candidate IDs A and B for election 1 and candidate IDs C and D for election 2. This simple candidate numbering solves the problem of simultaneous elections but can lead to the large candidate list issue.

Next, we present the details of the Matrix CodeVoting, an enhancement to CodeVoting to support large candidate lists and, consequently, simultaneous elections as previously explained. Additionally, it also provides better security in the case of reuse of a CodeCard in consecutive elections. The Matrix CodeVoting allows the use of the CodeVoting technique in elections with a large candidate list by using an encryption matrix that stores a large number of candidate ID transformations in a compact form.

5.1 Matrix CodeVoting Details

The Matrix CodeVoting replaces the original CodeCard by a Matrix CodeCard (Fig. 4). The ballot format suffers minor changes and the candidates are identified by numbers (e.g. 54598, 39027, etc.) instead of letters (e.g. A, B, C, etc.), as illustrated in Fig. 5. The Matrix CodeCard consists in a matrix that the voter will use to translate (encrypt) the candidate ID to the vote code, in opposition to the direct associations printed in the original CodeCard.

Figure 6 shows how to use the Matrix CodeCard to get a vote code from a candidate ID. In this example we use a 5-digit candidate ID which supports 100000 different candidate IDs, a much larger number than the number of possible candidates with the first CodeCard design. With the help of the Matrix CodeCard the voter translates the candidate ID into a character string.

Of course, we still need to be careful when selecting the candidate IDs. For instance, if we have nine candidates it is a bad idea to use candidate IDs from

Matrix CodeCard

Candidate number	d_5	d_4	d_3	d_2	d_1
0	A	S	Q	B	U
1	W	E	P	S	I
2	E	W	L	V	R
3	R	Q	I	G	S
4	Y	U	M	N	J
5	V	N	H	Y	H
6	B	J	T	U	T
7	M	M	F	K	C
8	O	X	E	W	E
9	U	T	Z	A	P
Vote code	v_5	v_4	v_3	v_2	v_1

Encoding Matrix

Confirmed vote delivery code: 6HKG2

Fig. 4. Matrix CodeCard for 5-digit candidate numbers

Election for the most important figure in security.

A - Alice
B - Bob
C - Eavesdropper
D - Attacker

Enter your vote code:

Election for the most important figure in security.

54598 - Alice
39027 - Bob
78351 - Eavesdropper
46209 - Attacker

Enter your vote code:

Fig. 5. A regular ballot is presented on the left and the Matrix Code-Voting is shown on the right

Candidate number	4	6	2	0	9
0	A	S	Q	**B**	U
1	W	E	P	S	I
2	E	W	**L**	V	R
3	R	Q	I	G	S
4	**Y**	U	M	N	J
5	V	N	H	Y	H
6	B	**J**	T	U	T
7	M	M	F	K	C
8	O	X	E	W	E
9	U	T	Z	A	**P**
Vote code	Y	J	L	B	P

Encoding Matrix

Fig. 6. Example of the encryption of a candidate ID using the Matrix CodeCard

00001 to 00009. In this case, the vote code for all candidates only differs in the last digit; therefore, an attacker willing to change the voter's vote to another candidate would have a *Probability of Guessing a Valid Vote Code* $Pgvvc = \frac{8}{25}$, i.e. 32% of success in the first try. This probability can be reduced by increasing the domain of the vote codes, e.g. by using capital and non-capital letters. With this example it is clear that the candidate IDs should be as different as possible to minimize the probability of an attacker guessing a valid vote code from the one typed in by the voter, e.g. 02536, 63875 and other IDs where all digits in the same positions are different.

Table 1. Number of different candidate IDs for $4 \leq k \leq 6$ and candidate ID length from 5 to 8. The results are an average of 100 rounds of random code generation.

	Candidate ID Length			
Security Parameter	5	6	7	8
$k = 4$	68	502	3483	25014
$k = 5$	-	65	387	2380
$k = 6$	-	-	58	298

Since the candidate IDs are in base 10 it is only possible to have 10 candidate IDs that differ in all positions. We propose the adoption of a security parameter k, which defines the number of different digits between all candidate IDs. The value of k must be chosen taking into account the probability of an attacker guessing, at the first try, a valid vote code from the one inserted by the voter. i.e. $Pgvvc = (\frac{1}{numberOfCodeSymbols-1})^k$.

In Table 1 we present the possible number of candidates for $4 \leq k \leq 6$ and a candidate ID length from 5 to 8. Table 1 shows that the Matrix CodeVoting is able to securely support large candidate list elections by slightly increasing the length of candidate IDs.

As explained before, the problem of simultaneous elections can be reduced to a large candidate list problem. One hasty conclusion is to assume that if the Matrix CodeVoting offers a solution for elections with a large candidate list it can be used to securely conduct simultaneous elections. We must say that this is not entirely correct because if an attacker has access to different vote codes, generated with the help of the same Matrix CodeCard, it can correlate the codes, discover parts of the matrix and possibly substitute the voter's vote.

The minimum number of codes needed to expose the entire matrix is 11, i.e. if a unique correspondence between the candidate numbers can be established only through an analysis of the repeated digits in the codes. For instance, given two simultaneous elections with the following pairs of candidates 12345, 67890 and 43567, 82059. A voter could vote for candidate 12345 in the first election and for candidate 82059 in the second election. Since the selected candidates are the only ones that share the second digit, the vote client application automatically has access to nine encrypted symbols in the encoding matrix, one in the column of the shared digit and two in all the other columns.

To minimize this threat it is important to choose candidate IDs in a way that no repeated digits in the vote codes of different elections can allow for a unique correspondence between candidate IDs, i.e. making the vote code correlation harder by adding confusion to the process.

Concluding, the Matrix CodeCard reuse should be limited to a few elections to prevent successful correlation attacks.

6 Conclusions and Future Work

Cryptographic voting protocols protect voting from manipulation at server side, if the protocol is executed by a trusted vote client. However, in remote voting, e.g. Internet voting, the voters use uncontrolled platforms to vote, therefore there are no guarantees that the client machine is trusted.

In this paper we presented the CodeVoting technique that protects the voter's vote from malicious manipulations at the uncontrolled/untrusted vote client platform, e.g. the voter's PC. CodeVoting uses a "code sheet" approach to protect the communication between a voter and a VST attached to the voter's PC. The VST runs a cryptographic voting protocol to protect the vote from manipulation at server side.

We presented the first code sheet format called CodeCard, that enables a secure communication between the voter and her/his VST. However, this format does not support elections with large candidate lists. Additionally, for improved security, the voter should renew her/his CodeCard between elections. Then, we introduced the Matrix CodeCard, a new CodeCard design that allows the support of large candidate lists. The use of the Matrix CodeCard in consecutive elections is also more secure.

In the future we want to enhance CodeVoting by introducing support for multi-candidate selection and ordering.

At the moment, we have working prototypes of the CodeCard and the Matrix CodeCard implemented in smart cards, namely JavaCards. We are now working on the integration of the CodeCard prototypes with a fully cryptographic vote client inside the VST (JavaCard).

Acknowledgments

We thank Patricia Lima and the editors of this book for the careful reading and suggestions made to improve the legibility of this manuscript.

References

1. CERT: Vulnerability remediation statistics (2007), `http://www.cert.org/stats/vulnerability_remediation.html`
2. USCERT: Cyber security bulletins (2007), `http://www.us-cert.gov/cas/bulletins/`

3. Wikipedia: Pharming (2007), http://en.wikipedia.org/wiki/Pharming
4. Stamm, S., Ramzan, Z., Jakobsson, M.: Drive-by pharming (2006), http://www.symantec.com/avcenter/reference/Driveby_Pharming.pdf
5. Gaudin, S.: Pharming attack slams 65 financial targets. InformationWeek (2007), http://www.informationweek.com/showArticle.jhtml?articleID=197008230
6. Kirk, J.: Pharming attack hits 50 banks. IDG News Service, TechWorld (2007), http://www.techworld.com/security/news/index.cfm?newsid=8102
7. Council of Europe: Family voting. Congress of Local and Regional Authorities of Europe session (2002), http://www.coe.int/T/E/Com/Files/CLRAE-Sessions/2002-06-Session/family_voting.asp
8. Volkamer, M., Grimm, R.: Multiple casts in online voting: Analyzing chances. In: Robert Krimmer, R. (ed.) Electronic Voting 2006, Castle Hofen, Bregenz, Austria. LNI, vol. P-86, pp. 97–106. GI (2006)
9. California Internet Task Force: A report on the feasibility of internet voting (2000), http://www.ss.ca.gov/executive/ivote
10. Internet Policy Institute: Report of the national workshop on internet voting: Issues and research agenda (2001), http://www.diggov.org/archive/library/dgo2000/dir/PDF/vote.pdf
11. Jefferson, D., Rubin, A.D., Simons, B., Wagner, D.: A security analysis of the secure electronic registration and voting experiment (serve) (2004), http://www.servesecurityreport.org/paper.pdf
12. Rivest, R.L.: Electronic voting. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, p. 243. Springer, Heidelberg (2001)
13. Rubin, A.D.: Security considerations for remote electronic voting. Commun. ACM 45(12), 39–44 (2002)
14. Joaquim, R., Ribeiro, C.: Codevoting: protecting against malicious vote manipulation at the voter's pc. In: Chaum, D., Kutyłowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) Frontiers of Electronic Voting, no. 07311 in Dagstuhl, Germany. Dagstuhl Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
15. Joaquim, R., Ribeiro, C.: CodeVoting protection against automatic vote manipulation in an uncontrolled environment. In: Alkassar, A., Volkamer, M. (eds.) VOTE-ID 2007. LNCS, vol. 4896, pp. 178–188. Springer, Heidelberg (2007)
16. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 244–251. Springer, Heidelberg (1993)
17. Joaquim, R., Zúquete, A., Ferreira, P.: Revs - a robust electronic voting system (extended). IADIS International Journal of WWW/Internet 1(2), 47–63 (2003)
18. Ohkubo, M., Miura, F., Abe, M., Fujioka, A., Okamoto, T.: An improvement on a practical secret voting scheme. In: Zheng, Y., Mambo, M. (eds.) ISW 1999. LNCS, vol. 1729, pp. 225–234. Springer, Heidelberg (1999)
19. Okamoto, T.: Receipt-free electronic voting schemes for large scale elections. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 25–35. Springer, Heidelberg (1998)
20. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–88 (1981)

21. Clarkson, M., Chong, S., Myers, A.: Civitas: A secure remote voting system. In: Chaum, D., Kutylowski, M., Rivest, R.L., Ryan, P.Y.A. (eds.) Frontiers of Electronic Voting, Dagstuhl, Germany. Dagstuhl no. 07311 in Seminar Proceedings, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2007)
22. Neff, C.A.: Verifiable mixing (shuffling) of elgamal pairs (2004), http://votehere.com/vhti/documentation/egshuf-2.0.3638.pdf
23. Park, C.-s., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/Nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994)
24. Benaloh, J.C.: Verifiable Secret-Ballot Elections. PhD thesis, Yale University (1987)
25. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997)
26. Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
27. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 539–556. Springer, Heidelberg (2000)
28. Estonian National Electoral Commitee: Internet voting in estonia (2007), http://www.vvk.ee/engindex.html
29. Lee, B., Kim, K.: Receipt-free electronic voting scheme with a tamper-resistant randomizer. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 389–406. Springer, Heidelberg (2003)
30. Oppliger, R.: How to address the secure platform problem for remote internet voting. In: Erasim, E., Karagiannis, D. (eds.) 5th Conference on "Sicherheit in Informationssystemen" (SIS 2002), Vienna, Austria, pp. 153–173. vdf Hochschulverlag (2002)
31. Zúquete, A., Costa, C., Rom ao, M.: An intrusion-tolerant e-voting client system. In: 1st Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS 2007), Lisbon, Portugal (2007)
32. TGC: Trusted computing group (2007), https://www.trustedcomputinggroup.org/home
33. Sadeghi, A.R., Selhorst, M., Stüble, C., Wachsmann, C., Winandy, M.: Tcg inside?: a note on tpm specification compliance. In: STC 2006: Proceedings of the first ACM workshop on Scalable trusted computing, Alexandria, Virginia, USA, pp. 47–56. ACM, New York (2006)
34. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Pftzmann, B., Liu, P. (eds.) CCS 2004: Proceedings of the 11th ACM conference on Computer and Communications Security, Washington DC, USA, pp. 132–145. ACM, New York (2004)
35. Volkamer, M., Alkassar, A., Sadeghi, A.R., Schulz, S.: Enabling the application of the open systems like pcs for online voting. In: Frontiers in Electronic Elections Workshop (FEE 2006), Hamburg, Germany (2006)
36. Chaum, D.: Surevote (2001) International patent WO 01/55940 A1, http://www.surevote.com/home.html
37. UK's Electoral Commission: Technical report on the may 2003 pilots (2003), http://www.electoralcommission.org.uk/about-us/03pilotscheme.cfm

38. UK's National Technical Authority for Information Assurance: e-voting security study (2002), http://www.ictparliament.org/CDTunisi/ict_compendium/paesi/uk/uk54.pdf
39. Helbach, J., Schwenk, J.: Secure internet voting with code sheets. In: Alkassar, A., Volkamer, M. (eds.) VOTE-ID 2007. LNCS, vol. 4896, pp. 166–177. Springer, Heidelberg (2007)
40. Kutyłowski, M., Zagórski, F.: Verifiable internet voting solving secure platform problem. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 199–213. Springer, Heidelberg (2007)
41. Skagestein, G., Haug, A.V., Nødtvedt, E., Rossebø, J.E.Y.: How to create trust in electronic voting over an untrusted platform. In: Krimmer, R. (ed.) Electronic Voting 2006, Castle Hofen, Bregenz, Austria. LNI, vol. P-86, pp. 107–116. GI (2006)