The Secure Platform Problem
Taxonomy and Analysis of Existing Proposals to Address this Problem

Michael Schläpfer
ETH Zurich
Universitätsstrasse 6
CH-8092 Zurich
+41 44 632 91 96
michschl@inf.ethz.ch

Melanie Volkamer
Technische Universität Darmstadt
Hochschulstrasse 10
64289 Darmstadt, Germany
+49 6151 16 5422
melanie.volkamer@cased.de

ABSTRACT
One of the main open issues in electronic government is the fact that the individual users' multi-purpose computing platforms are used. In terms of security, no guarantee is given since these platforms are not under the government authority's control. Even worse, the number of malware infected computing platforms increases. This so-called Secure Platform Problem and approaches aiming to solve it are objects of investigation in this work. We define criteria that need to be ensured to address this problem. Furthermore, we propose a taxonomy to classify existing approaches. Based on the classification and the criteria, we analyze the different types of approaches by providing concrete examples. Hereby, we show that none of the existing approaches fully meets our criteria. Thereby, we focus on the most security critical class of electronic government services, namely electronic voting over the Internet. However, most of the discussed approaches as well as the criteria and classification can also be applied to other governmental applications.

Categories and Subject Descriptors
K.6.5 [MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS]: Security and Protection – Authentication, Invasive Software, Physical Security, Unauthorized Access

General Terms
Security, Human Factors, Design, Reliability, Verification

Keywords
Electronic Government, Internet voting, Secure Platform Problem, Insecure Client Problem, Untrusted Terminal Problem

1. INTRODUCTION
Web applications become more and more important and a growing number of security-critical services are provided over the Internet. The latest advances are, to provide citizens with governmental services over the Internet, even with online voting techniques for political elections and referenda what we refer to as *internet voting* throughout this paper. One of the main open issues of electronic government is that the individual user's multi-purpose computing platform is used as the interface to the service. Since this platform is not under the authority's control, no security guarantees can be given. Even worse, the increasing complexity of the operating systems leads to a growing number of malware indicating that it is reasonable to not trust any user's personal computer at all. The same applies to mobile computing platforms such as mobile phones and tablet computers. In the context of internet voting, this so-called *Secure Platform Problem* was first mentioned by Ron Rivest [24]. The problem is also known as the *Untrusted Terminal Problem* or the *Insecure Client Problem* and is not voting specific but applies to any electronic communication application. However, compared to other domains, the problem appears to be most influential in electronic government services and especially in internet voting. Vulnerabilities in voting systems endanger democracy in its entirety. Furthermore, compared to other applications, voters have to be anonymous in order to ensure ballot secrecy.

Many voting protocols, such as [5], [15], and [7] have been proposed in the last thirty years and some have already been used for legally binding elections on different levels. Examples include Estonia, Switzerland, and the Netherlands (compare to [28] for an overview of all three elections). But most of these voting protocols and systems in use do not even mention the Secure Platform Problem, while others trivially assume the voter's equipment to be trustworthy, arguing "up-to-date" operating systems and anti-virus programs, as well as vote updating being viable solutions to address the problem.

However, there exist few publications proposing solutions to (partially) address the Secure Platform Problem in internet voting [19][32]. In addition, several solutions for user and transaction authentication in e-banking are available which can be adapted for governmental web services like internet voting as we will show. The fact that these proposed and existing techniques are rarely applied in governmental web services, yields that the people in charge are either not aware of the Secure Platform Problem or the approaches are not known enough, yet. This is not surprising as there is no comprehensive document describing the different

ICEGOV '12, October 22 - 25 2012, Albany, NY, USA

concepts including their advantages and disadvantages as well as the remaining risks.

This work aims to narrow this gap. We present a taxonomy to categorize existing approaches and define criteria, an adequate solution for the Secure Platform Problem should meet. Furthermore, we provide an overview of existing proposals to address the Secure Platform Problem proposed for different applications and analyze them according to the previously defined criteria.

Furthermore, we intend to raise awareness for this problem that is mostly neglected and to support decision makers in electronic government projects with an overview of the currently most relevant approaches to meet the Secure Platform Problem including their limitations. Additionally it should encourage developers to integrate one or more of these approaches in their products, and researchers to find more adequate solutions.

1.1 Related Work

To our knowledge, no comprehensive overview and discussion of solution approaches for the Secure Platform Problem exists. However there are few works addressing subsets of existing approaches.

The author of [32] describes and compares five different approaches to address the problem. The discussion is brief and does only address a fraction of the approaches discussed in this work.

[19] provides a good introduction and overview of the Secure Platform Problem. The author classifies some meaningful approaches and concludes that dedicated hardware devices can provide the highest security level against the Secure Platform Problem while he does not analyze this branch of solutions in any detail due to the expected costs for development, distribution and maintenance.

The common conclusion of the above authors is that none of the existing approaches is applicable for large-scale elections.

1.2 Structure of this Work

Section 2 provides an overview of the Secure Platform Problem. We introduce a new informal adversary model and express the adversary's capabilities. In Sect. 3 we formulate the criteria we used to discuss the different existing solution approaches as well as a simple taxonomy, which serves to classify the approaches in Sects. 4 – 6. Section 7 concludes this work and summarizes some open issues for future work.

2. PROBLEM DESCRIPTION

In this section, we explain the Secure Platform Problem and define a new adequate adversary model.

In the context of security critical communication an adversary's ultimate goal is to violate confidentiality and/or integrity of messages sent and received by users. For the purpose of this paper we define a powerful adversary capable of entirely controlling the platform of the user. Thus, the adversary does not only control the network between the user's computing platform and the authority's trusted server, but also acts as the user's interface to the network. We abstract from the untrusted platform and assume it to be part of the adversary-controlled network. Communication takes place between the user and the adversary as well as between the server and the adversary. Hence, all communication between the user and the server is sent through the adversary. Note that the user is not a machine but human and thus is very limited in terms of computation power and memory. Fig. 1 depicts our adversary model. This model extends the common formal protocol analysis models based on the well-known Dolev-Yao-style adversary [8].

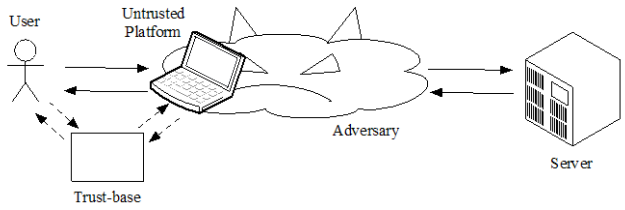

Figure 1. Secure Platform Problem Adversary Model

We add a trust-base to our model, in order to address those approaches, which introduce techniques to execute certain trustworthy functionality outside the influence of the adversary. The functionality and especially the communication capabilities of the trust-base depend on the individual approaches as well as on the respective implementation details.

Since we focus on the client-side Secure Platform Problem only, we assume the server-side as well as the user to be trustworthy. More specifically, we do not consider the case where users intend to collaborate with the adversary as it would be the case for vote selling in internet voting. Furthermore we do not take into account the adversary to be physically present to influence the user.

Our adversary model yields the following capabilities. We recap the fundamental capabilities of the Dolev-Yao adversary and extend them with human capabilities. In particular we assume that the adversary:

- can learn messages sent from the user to the untrusted platform;
- can learn messages sent from the server to the network and further to an untrusted platform;
- can learn messages sent from the trust-base to an untrusted platform or directly to the network;
- can drop messages to replace them with own messages[1];
- can manipulate and fabricate arbitrary messages according to publicly known knowledge and previously sent messages;
- can perform every publicly known function;
- knows all implementation details of all used systems;
- can act as a human[2].

Although the above assumptions describe a powerful adversary, the capabilities are limited. In particular, the adversary:

[1] We do not consider denial-of-service attacks in this work.

[2] Note, for example, that strategies against Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHAs) exist.

- cannot break cryptography;
- cannot overhear or manipulate communication between the user and the trust-base;
- cannot access knowledge dedicated to the user, the trust-base or the server;
- cannot manipulate the trust-base.

3. EXISTING APPROACHES

In this section, we introduce the properties that will be used to later on discuss and compare existing approaches. While we are mainly interested in confidentiality and integrity of messages, we settle for the following more precise definitions and describe how they are related. Note that the following criteria and taxonomy hold for security-critical systems in general. However, we will take internet voting and corresponding solutions as an example to discuss the different approach classes in Sects. 4 – 6.

3.1 Criteria

While we are mainly interested in confidentiality and integrity of messages, we settle for the following more precise definitions and describe how they are related.

3.1.1 Confidentiality

An approach supports *user-to-server-secrecy* if it is not possible for the adversary to learn a secret that the user submitted to server. This means that the user has the opportunity to submit a message secretly to the authority's server and hence privacy holds.

3.1.2 Integrity

If it is possible for the server to verify the integrity and authenticity of a received message with respect to the user-sent message, we say the approach supports *user-to-server-integrity*. Note that individual verifiability in internet voting allows the voter to verify that the server correctly received the voter's ballot. But this is not enough to ensure vote integrity since the voter must also be given the possibility to complain in the case the verification fails. Moreover, individual verifiability also requires integrity and authenticity of messages sent from the server to the user in order for the user to be able to verify what message the server actually received. From the authority's perspective, vote integrity is then given by the fact that the user did not complain.

3.1.3 Further Criteria

Especially but not exclusively in internet voting, an application should provide anonymity. We say an approach supports user-anonymity if it is not possible for the adversary to reveal the origin of a message sent to the server. Note that it is possible for a user to send messages that are not secret but cannot be linked to the sender and thus privacy holds too.

Beside security properties other non-functional properties such as user-friendliness, cost-efficiency, and practicality for large-scale settings must be considered as well. Although we focus on security, we also discuss these properties where it applies but we do not define any measurement for that.

3.2 Taxonomy of Approaches

Following, we propose a taxonomy to give an overview and to classify the different approaches. In scientific literature mainly two different classes of solutions for the Secure Platform Problem can be found. While one class aims to make the platform trustworthy, the other class of approaches assumes the user's platform to be insecure. See Fig. 2 for an overview of the classes that are covered in this work.

In the following sections we refer to this taxonomy, summarize concrete examples, and discuss them according to the above criteria. Particularly, Sect. 4 examines approaches to make the platform trustworthy. Approaches distrusting the user's platform are discussed in Sect. 5, where we summarize approaches without trusted devices and Sect. 6, where we discuss solutions based on dedicated trusted devices.

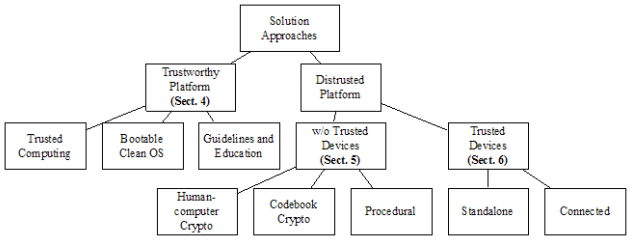

Figure 2. Taxonomy of Analyzed Approaches

4. TRUSTWORTHY PLATFORM

Approaches in this category aim to improve the user's platform integrity in a way that it becomes trustworthy. These approaches lead to the situation where the platform becomes the trust-base and users no longer access the communication interface to the adversary.

4.1 Trusted Computing

The key idea of applying Trusted Computing [29] techniques to overcome the Secure Platform Problem is to use an appropriate security architecture based on a security kernel and special Trusted Computing hardware. [1] and [33] discuss the application of this approach for internet voting in detail.

Trusted Computing efficiently overcomes malicious software on the user's computing platform because an eligible message can only be created after successful verification of the system's integrity. However, there are still open problems with Trusted Computing, such as in remote attestation [31]. Moreover, the concept is not widespread enough, users might already have a personal computer with integrated Trusted Platform Module but the security architecture and security kernel are still missing. It is questionable whether or not this technique will be applied on a large scale in near future. However, the approach would allow convenient and user-friendly solutions for secure services, even for internet voting as this would all run in the background.

4.2 Bootable Clean Operating System

Otten [22] recommended developing a special voting operating system based on an open-source operating system that boots and runs directly from a read-only data medium such as a CD or DVD. This medium would then be distributed to all users. After having

received, users need to configure their computer to boot from this medium. Additional security checks and secure distribution are required to overcome the risk of getting a malicious medium that, for example, communicates with a malicious server.

While this approach overcomes many Secure Platform Problem related issues, it does not meet the particular problem of a manipulated BIOS. Such an attack allows the adversary to load a malicious environment in which the secure operating system is virtually executed without the user or the authority noticing it. Avoiding this kind of attack is only possible by applying Trusted Computing hardware as described before. Therefore, it does not ensure secrecy, nor integrity in our adversary model. However one has to keep in mind that developing such low-level attacks is not a trivial task and an adversary would have to cope with a variety of different hardware settings to successfully mount a large-scale attack in such a setting. Challenges include developing a CD or DVD that boots from all the different hardware and software settings around and that all necessary drivers are provided to be able to automatically connect to the Internet. This may lead to high development, distribution, and maintenance costs. Another difficulty for the authority is to verify that users really used the clean operating system and that it was running on hardware, not in a virtual machine potentially under the adversary's control.

4.3 Guidelines and Education

A simple approach to address the Secure Platform Problem is the provision of special guidelines and the education of users in how to protect their own computer systems from malware. Examples include the guidelines provided for the student elections in Austria [30] and those developed by the German society of computer scientists [11]. They include information about software updating, firewall-settings and how to verify SSL certificates.

This approach claims to reduce the probability that malware infects a user's system. However, only standard and well-known attacks can be prevented and it is questionable whether a sufficient fraction of users are able to follow the guidelines and to really protect their systems. In addition, users cannot be forced to apply the security guidelines. Regarding user-friendliness, it is likely that many users would not follow the instructions because of additional work. Note that in contrast to e-banking the laziness of an individual user in e-voting not only impacts herself but the outcome of the election, i.e., the entirety of the participating users is affected. Furthermore, an adversary could distribute modified guidelines to mislead inexperienced users to behave incorrectly and thus to put themselves at risk.

5. WITHOUT TRUSTED DEVICES

Distrusting the user's computing platform leads to the need of establishing a secure channel directly between the user and the server. This channel can either be established using cryptography or by an out-of-band channel that is not under the adversary's control.

5.1 Human-Computer Cryptography

Human-computer or *paper-and-pencil* cryptography has a long history and given enough time, pencils, and paper, humans may theoretically perform every computation a computer can do. In terms of our model, such approaches aim to provide the user with the capabilities to encrypt and authenticate messages directly. Hence, a trust-base is not needed and all the defined security properties can be achieved. However, the vast majority of humans lack sufficient memory and computation power to perform strong cryptographic operations in a reasonable amount of time. Nevertheless, some interesting approaches such as [16] and Schneier's Solitaire algorithm [27] were proposed. For example, in Solitaire the randomness in a shuffled deck of playing cards is used for the encryption of messages but the approach has some weaknesses.

Bertà examined in [3] the human limitations with respect to encrypting and authenticating sufficiently large messages in practical settings. He concludes that no sufficiently strong cryptographic protocol to encrypt or authenticate messages exists such that humans could apply it.

Because of the obvious lack of user-friendliness we do not consider human-computer cryptography as a practical solution for the Secure Platform Problem.

5.2 Codebook Cryptography

Although humans are not good in performing reasonably strong cryptographic operations, they are considerably strong in comparing patterns. This fact is exploited by the concept of codebook cryptography. The idea is that prior to the communication, the trusted server encrypts all possible messages and then distributes the corresponding codebook, i.e., the clear-text / cipher-text mapping, over an out-of-band channel[3] to the user. The user chooses a clear-text message and looks up the corresponding cipher-text (code) to be entered into the untrusted platform. The platform then sends the cipher-text to the server. Hence, the adversary does not learn the clear-text message and since not in possession of the codebook, the adversary cannot replace the message with another one that is accepted by the server. Thus, this approach is resilient against the Secure Platform Problem but needs a secure out-of-band channel for distribution.

Following we present some widely discussed examples of this approach for electronic voting.

5.2.1 Code Voting

Chaum was the first who applied this concept to internet voting [5]. The key idea of Code Voting is that the user gets a code sheet together with the general election information via mail. The code sheet links each candidate or party to a random alphanumeric code. In order to cast a vote for a specific candidate, the user enters the corresponding code into a text-field. This code is submitted to the authority that is able to derive the user's choice by mapping the code back to the candidate. Since not in possession of the code sheet, the untrusted platform (i.e., the adversary) can neither derive the user's choice nor change the ballot's content to another meaningful choice. Helbach and Schwenk [14] as well as Oppliger et al. [20] proposed different improvements like additional verification codes that are sent back from the server for individual verifiability. To overcome the vote-buying problem the authors in [21] introduced an additional finalization code. Ryan et al. recently suggested in [26] and [13]

[3] In this case the trust-base represents the out-of-band channel that provides confidentiality and integrity of the received codebook.

to use Code Voting in combination with Prêt-à-Voter in order to allow individual verifiability. In this approach, the server can only send a valid confirmation code to the user if a majority of trustees confirm the correct recording of the voting code that was sent by the user. The main disadvantage of the Code Voting approach concerns user-friendliness, which decreases in particular for implementing complex ballots with multiple candidates to choose from. In addition, a trusted procedure to generate and distribute the code sheets is required what introduces additional complexity and costs.

5.2.2 Prêt-à-Voter

Another elegant implementation of codebook cryptography can be found in Prêt-à-Voter [25], where the candidate list is permuted and the permutation is encrypted. While Prêt-à-Voter was proposed for poll-site-based settings, it could be adapted for internet voting too. The user receives the ballot paper with the permuted list of candidates and the encryption of the permutation via mail. Then, she enters the index of the chosen candidate together with the encryption of the permutation (which could also serve as a ballot sequence number). The permutation is decrypted by the authority's server and is used to derive the user's choice. However, since using a permuted list of valid choices, the platform, although not able to learn the user's choice, can mount a randomization attack by choosing a different index and therefore user-to-server-integrity is not provided while secrecy holds.

5.2.3 CAPTCHA

In [21] the authors propose the application of visual CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) to overcome the Secure Platform Problem. Candidates are displayed in a random order and represented by visual CAPTCHAs. This approach is simple to be integrated in any internet voting system. Its user-friendliness is questionable. While people might know this kind of images from other applications, it is often difficult to figure out what is displayed. Besides the fact that our model assumes an adversary with human capabilities, who is able to solve CAPTCHAs, it is particularly easy for the adversary to perform a randomization attack by just randomly choose a CAPTCHA different from the user-chosen one. A more advanced approach is discussed in [23]. Here, arbitrary CAPTCHAs, which are not related to the candidates' names, appear next to the actual candidate's name, while the candidate order can stay as it is. The user decodes the CAPTCHA related to her favorite candidate and enters the resulting code into a text-field. Automated breaking of these CAPTCHAs becomes more difficult because of the fact that the adversary does not know the encoded clear-text in advance. However, it still does not solve the Secure Platform Problem in our adversary model.

5.3 Procedural Approaches

Some approaches suggest adapting the voting process in order to overcome various problems in internet voting. Some of the proposals can even be applied to mitigate the Secure Platform Problem and are therefore discussed here.

5.3.1 Vote-Updating

The general idea of vote-updating is that the user can update the electronic vote as often as she wants, ideally from different devices. In hybrid systems, the user can even replace the electronic vote on election day by a traditional paper ballot at the polling station. There are several different approaches to enable vote-updating, while they have different advantages and disadvantages as shown in [12].

Enabling vote-updating, the adversary looses certainty about how the user really voted since she may have used another device or went to the polling station on election day to cast a different vote. Correspondingly, the adversary will not know for sure whether the modifications will influence the result or not. Of course, this only applies if the votes cannot be linked to the users.

Vote-updating helps particularly in the case where users distrust the voting system after casting their vote. For instance, this is the case when they misinterpret information presented to them, detect malware on their computers, or if they are not convinced that the ballot was properly sent and stored. However, statistics in Estonia, where internet voting, is used for political elections since 2007, indicate that only few people update their vote [9]. It might be the case that very few incidents happen but more likely most users are simply not able to notice manipulations and blindly trust their systems.

Vote-updating is easy to implement and to understand by users. However, a challenge is to ensure that only one vote is counted and no problems with replay-attacks or delays on the network can cause that an earlier cast valid vote is counted. Opponents of vote-updating argue that this approach influences the value and character of an election. They argue that the act of casting a vote is something special and should not be repeatable because otherwise it might get the character of a game.

5.3.2 Anonymous Voting

Anonymous voting allows the user to cast a ballot without the adversary being able to link the ballot to its origin. The following protocol serves as simple example for anonymous voting. Prior to the election the user gets an official letter from the election authority, containing credentials that allow the person in possession of these credentials to cast exactly one eligible ballot and the authority does not know who got what credential. Now the user uses a public computer in an internet cafe that cannot be associated to her and enters the received credentials to prove eligibility. Then she enters the favorite candidate and submits the vote together with the credentials to the election authority's server. The adversary learns the candidate whom was voted for but does not learn who voted. Hence, voter privacy holds. It is obvious that in this simple example neither secrecy nor integrity holds, unless combined with other approaches. A more advanced example for this category is based on blind signature schemes that have been proposed for the first time in [10] and was later used in many other protocol proposals. While this approach is more sophisticated from a cryptographic point of view, it does address the Secure Platform Problem to the same extent as the simple protocol above. Under the assumption that the untrusted system does not know the identity of the user who is using it, user-anonymity is given but neither secrecy, nor integrity.

5.3.3 Test Ballots

Test ballots are dedicated ballots the adversary cannot distinguish from real ones. The user casts one real ballot and some test ballots

in random order. Since the adversary does not know the real ballot he may manipulate one or more test ballots. After the election, the processing of all test ballots is made fully transparent to a group of auditors or even to the public for verification. Successful attacks require the adversary to manipulate a large number of votes and therefore it is likely to detect at least a fraction of the manipulations. The approach does not provide integrity of the real vote itself but strongly indicates possible manipulations by analyzing the proper handling of the test ballots. To detect manipulations in the context of the Secure Platform Problem, the test ballots must be individually verifiable by the user. Hence, the test ballots are exposed after the election's vote casting phase and the adversary learns the real ballot.

6. TRUSTED DEVICES

In the context of e-banking, a variety of technical solutions for user and transaction authentication have been presented and even deployed on a large scale. In the following we will analyze these approaches regarding the security criteria that we defined and give some hints on how such approaches could be adapted to internet voting.

Solutions based on *trusted devices* can be classified into standalone approaches, where the trusted device (TD) is not attached to the user's untrusted platform, and connected approaches, where the trusted device is connected to the adversary, for example, to the user's platform. The term trusted indicates that users must trust the devices to offer trustworthy functionalities and that they securely store confidential data. If the device additionally stores a specific user's secret information, such as cryptographic keys, we call the device a *personal trusted device* (PTD). Examples of personal trusted devices are smart cards. They can perform certain cryptographic operations and store individual secret information, i.e., the user's secret key. Since additional devices are required to access the smart cards, users need to trust these devices too. Hence, in smart card based approaches the card readers are also part of the trusted device. In our model, the trust-base is the user's smart card including the reader that is used to access the card.

6.1 Standalone Trusted Devices

Standalone trusted devices could be seen as calculators. The user communicates directly with the device and no communication takes place between the trusted device and any other party than the user.

Challenge-response authentication protocols for e-banking solutions that use a smart card reader with a pin pad and a display are an example for this category of approaches. The bank first sends a challenge code through the user's untrusted platform to the user who in turn enters this code into the card reader together with the bankcard's PIN (personal identification number) for accessing the card. The card then computes a MAC (message authentication code)[4] of the challenge code and secret information stored on the card. This MAC is displayed to the user through the reader's display. Finally, the user enters the MAC into the untrusted platform and submits it back to the bank. If the response code corresponds to a valid MAC, the bank is convinced that the correct card (containing the secret information) as well as a person who knows the correct PIN to access the card are involved in the protocol run.

To adapt this approach to internet voting, such a device could be used to encrypt and authenticate the ballot. The user enters the candidate choice directly into the PTD and gets back an encrypted ballot, for example, represented as an alphanumeric code. The user enters the encrypted ballot into her untrusted platform and submits it to the server. Such an approach can be used to provide "digital" code voting without a separated out-of-band channel. The initial vote codes are digitally provided to the users via their untrusted platforms. The vote codes are then used like the challenges in the above described challenge-response authentication example. The user enters the MAC of a secret key and the vote code that corresponds to her choice into the untrusted platform to submit this vote to the server. The authority in possession of the same secret key computes the MACs for all candidate codes and compares them with the received MAC to derive the user's choice. Furthermore, the server could again compute the MAC of the received MAC and send it back to the user as a confirmation code that can be verified by the user when again performing the same operation using the PTD. In practice a more sophisticated scheme is required to prevent the authority from learning individual users' votes.

Since no clear-text referring to a user's choice is submitted to the adversary, the approach supports user-to-server-secrecy. The other way round the user can use the trust-base to decrypt messages encrypted by the server. The setting provides user-to-server-integrity if combined with a verifiable confirmation code as described above. However, user-friendliness is questionable since the user must relay all encrypted communication between the trust-base and the server. In the case of a few short codes, this may be applicable but not necessarily in complex elections where the user must choose multiple candidates.

6.2 Connected Trusted Devices

In contrast to standalone TDs, connected TDs are attached to the adversary, in most cases to the user's untrusted platform, thus there is a communication channel between the adversary and the trust-base. Simple approaches offer specific functionalities to the platform they are attached to such as securely storing secret information. More sophisticated networked devices are able to directly communicate with the remote server. Generally, the more functionalities a TD offers, the more vulnerabilities may occur. However, in our model we limited the adversary's capabilities by assuming that the adversary cannot manipulate the trust-base.

The communication interfaces of a connected TD directly affect the possible security guarantees of any protocol making use of it. To make this clear, we will summarize the possible communication capabilities of the trust-base, provide examples, and discuss the properties of each setting.

6.2.1 No Communication Between User and Trust-Base

The trust-base has no interface to communicate with the user and all communication is based on the untrusted platform (which is

[4] Other cryptographic operations can be used and in practice, asymmetric cryptography is often applied, where the challenge code is signed using the secret key stored on the card.

part of the adversary). Particularly the user can only communicate with the adversary and neither secrecy nor integrity can be achieved with respect to our model. The only guarantee in this setting is that during the protocol run, the TD is attached to a computer, which is reachable by the adversary.

Widely deployed examples for this setting are smart cards in combination with class 1 card readers directly attached to the platform via USB. Unfortunately, the adversary immediately learns every message sent by the user, particularly the PIN that protects access to the card. Hence, after the first time the user enters the PIN, the adversary may arbitrarily use the card and perform whatever cryptographic operation the smart card offers, as long as it is attached to the untrusted platform. Applying this to internet voting the user does not know what is being processed by the TD and in a setting where vote updating is allowed, the user does not even notice that the smart card is being used at a later time to replace a vote. Another example are hardware tokens like the mIDentity developed by Kobil [17] for e-banking and other sensitive applications where a browser runs in a "secure" environment on the TD. However, as there is no communication between the trust-base and the user none of the defined security properties are guaranteed with respect to our model.

6.2.2 Unidirectional Communication from the User to the Trust-Base

The TD provides an input interface to the user such as a pin pad and the user can send clear-text messages to the TD, which are encrypted afterwards. Hence, such approaches provide user-to-server-secrecy as well as user-to-server-integrity. Depending on the protocol, the user can even verify the correct reception of the message (individual verifiability). To do so, the user enters her choice together with a random number into the TD, which in turn encrypts these messages with a secret key as well as a hash value thereof. The secret key is shared between the server and the TD. The TD sends the encrypted values to the adversary for submitting them to the server. After reception, the server first decrypts them and verifies their integrity before sending the random number back to the adversary as confirmation. The adversary provides this confirmation to the user and the user is convinced that the server received the correct message if and only if the confirmation corresponds to the previously entered random number. Since the adversary cannot break cryptography, the best strategy would be to guess the random number, which is not feasible for large enough numbers. In this setting the server cannot send any secret or authentic message to the user because the TD is not able to communicate with the user differently than through the adversary.

Examples include smart cards in combination with class 2 readers. These readers provide a pin pad to prevent the platform from learning the PIN since it is directly entered into the trusted card reader. Because the adversary does not learn the PIN, unnoticed access to the smart card's functionality is prevented.

6.2.3 Unidirectional Communication from the Trust-Base to the User

The TD has an output interface such as a display or a speaker. The adversary learns every message sent by the user. Therefore A can learn every message sent by the user and again the adversary may learn the PIN to access TD's functionality. It is possible to avoid this, for example, by displaying a randomly permuted pin pad on the untrusted platform's display on which the user has to enter the PIN [4]. The corresponding permutation is displayed on the TD's display. Every time the TD is accessed it challenges the user, using the output interface. Moreover, the TD's output interface can be used to verify that the server received the correct message. To do so, the server sends confirmation messages, which the trust-base is able to decrypt and verify. The verification result is then provided to the user. Hence, depending on the concrete implementation, secrecy and integrity from the user to the server as well as vice versa can be achieved. Thus, it overcomes the Secure Platform Problem but requires sophisticated user interfaces to assure secrecy and therefore lacks user-friendliness for practical applications. User-to-server-integrity can conveniently be achieved by individual verifiability. More specifically, the PTD's trustworthy output interface can be used to verify information received by the server.

Examples are smart cards and readers with only a confirmation display. The display can be used to verify data that will be processed by the smart card. Another example. Which is already widely applied in e-banking is transaction authentication. Confirmation information of every transaction entered by the user is sent over a dedicated channel to the user. The user verifies the correctness and authorizes the transaction. One concrete implementation is based on short messages and mobile phones as TDs. The user enters the desired transaction into the untrusted platform and sends it to the bank's server. The server generates a transaction authentication number (TAN) and sends a short message containing the beneficiary's account number, the amount, and the TAN to the user's mobile phone. Now, the user verifies the correctness of the transaction information. If verifica- tion is successful, the user authorizes the transaction by entering the TAN into the untrusted platform to send it to the server. The server compares the received TAN with the expected TAN and accepts the transaction if they are equal. This concept could be used for internet voting to allow individual verification but it is only secure as long as the adversary cannot break confidentiality and integrity of the short messages sent by the authority's server. Unfortunately today's implementations cannot prevent either. Moreover, it is questionable whether it is reasonable to assume mobile phones to be trustworthy. In near future communication platforms may conflate and voting as well as short messaging may be performed using the same platform, which introduces the Secure Platform Problem again.

6.2.4 Bidirectional Communication between the Trust-Base and the User

The trust-base provides an input as well as an output interface to communicate with the user. In this setting the user may enter confidential messages directly into the TD. In contrast to the setting where communication exclusively takes place from the TD to the user, user-friendliness is improved because of the fact, that security-critical messages can directly be entered, thereby conveniently enabling user-to-server-secrecy. User-to-server-integrity is based on the ability of the TD to encrypt (and sign) messages. This setting also allows individual verifiability based on the TD's trustworthy output interface to the user.

Examples are smart cards in combination with class 3 readers. These devices offer a high degree of security and a convenient way for the user to securely communicate with the trusted server.

More complex solutions allow the TD to directly communicate with the server using standard network communication protocols. The Zone Trusted Information Channel (ZTIC) [2][34] was recently developed by IBM and is widely deployed by a popular Swiss bank. The ZTIC is a networked USB-attached smart card reader for application and transaction authentication in e-banking. The device consists of a small display and a few buttons. Transactions with yet unknown recipients have to be verified by the payer using the device's display and authorized by pressing a respective button on the device. The device communicates directly with the bank's server using the untrusted platform as a network proxy only. Adapting such a solution for internet voting offers a high degree of security while being convenient with respect to user-friendliness.

Borchert et al. propose the use of camera-equipped mobile phones for user and transaction authentication in e-banking. They provide a variety of ideas and implementations [4]. However, we do not consider mobile phones to be trustworthy for internet voting applications.

7. CONCLUSION AND FUTURE WORK

The Secure Platform Problem is one of the most serious problems in any online application and thus needs to be addressed in particular if it comes to e-governmental and internet voting applications. We introduced in this paper an adversary model that reflects the Secure Platform Problem as well as criteria which needs to be ensured in this model. We provided a taxonomy of approaches. We classified these approaches into those that aim to improve the trustworthiness of the user's platform and those that distrust the user's platform completely. We described and analyzed existing approaches to address the Secure Platform Problem in the context of remote electronic voting. The results can be summarized as followed:

Distrusting the platform requires a secure channel directly from the user to the election authority's server. This channel can be established either without or with trusted devices. We pointed out that approaches without trusted devices are more cost-efficient, while they also lack user-friendliness.

Trusted Computing offers convenient options for highly security critical applications such as internet voting. However, it is unclear whether corresponding operating systems and software will be available and used by a sufficiently large group of users in the foreseeable future. We noted that one must still trust the various manufacturers to have properly implemented the Trusted Computing hardware and software. For political systems where users participate only every couple of years, e.g., in elections, codebook cryptography seems to offer the most promising solution approaches. Trusted devices without communication interface to the user are useless regarding the Secure Platform Problem, more sophisticated solutions have great advantages with respect to user-friendliness as well as concerning the level of security obtainable. The high operational costs for maintenance, distribution, and support may not be in due proportion to the benefits for many voting settings unless the devices can be used for other applications too (e.g., e-banking or other frequently used e-government services). However, in countries like Switzerland where all citizens participate at political decisions several times a year, the use of special-purpose trusted devices might be applicable.

Thereby, we believe to raise awareness for the Secure Platform Problem that has been mostly neglected in past. We further believe that the results of this paper support decision makers in electronic government projects with an overview of the currently most relevant approaches to meet the Secure Platform Problem including their limitations. Additionally, we strongly believe that this paper encourage developers to integrate one or more of these approaches in their products, and researchers to find more adequate solutions.

Combinations of the discussed approaches may have effects on the security properties as well as on user-friendliness. An example is the combination of vote updating with codebook cryptography in internet voting as it is implemented in the Norwegian system [18]. Therefore, as future work we plan to analyze different combinations of existing approaches.

We exclusively focused on the client-side Secure Platform Problem and assumed the voter to be trustworthy. In specific settings like in internet voting this assumption does not necessarily hold. Therefore, we plan to examine existing approaches in the context of the entire voting system. Furthermore, it needs to be examined whether particular approaches impact verifiability or other important requirements which needs to be ensured in the context of internet voting. For instance, Code Voting requires a secure process for generating and distributing the code sheets since voter privacy must also hold against the authority. This will be part of future investigations, too.

8. ACKNOWLEDGEMENTS

We would like to thank Rolf Oppliger and the members of the Swiss E-Voting Competence Center for the fruitful discussions and valuable inputs. This research is partly supported by the Swiss Federal Chancellery.

9. REFERENCES

[1] A. Alkassar, A.R. Sadeghi, S. Schulz, and M. Volkamer. Towards Trustworthy Online Voting. *In Proceedings of the 1st Benelux Workshop on Information and System Security– WISSec*, 2006.

[2] M. Baentsch, P. Buhler, R. Hermann, F. Höring, T. Kramp, and T. Weigold. A Banking Server's Display on Your Key Chain. *ERCIM News*, pages 44–45, 2008.

[3] I.Z. Berta and I. Vajda. Limitations of humans when using malicious terminals. *Tatra Mountains Mathematical Publications*, 29:1–16, 2004.

[4] B. Borchert. Trojanersichere Online Accounts. *http://www-ti.informatik.uni-tuebingen.de/~borchert/Troja/*.

[5] David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM 24(2):* pages 84-88. 1981.

[6] D. Chaum. Surevote: technical overview. *In Proceedings of the Workshop on Trustworthy Elections (WOTE'01)*, 2001.

[7] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a secure voting system. *IEEE Symposium on Security and Privacy*, pages 354–368, May 2008.

[8] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198-208, 1983

[9] Estonia. Statistics of Internet Elections in Estonia. *http://www.vvk.ee/ voting-methods-in-estonia/engindex/statistics*.

[10] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. *In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology, ASIACRYPT '92*, pages 244– 251, London, UK, Springer, 1993.

[11] Gesellschaft für Informatik. Information für GI-Mitglieder zu möglichen Sicherheitsproblemen auf Client-seite bei Vorstands- und Präsidiumswahlen mit dem Onlinewahlverfahren. *Technical report, https://www.gi-ev.de/fileadmin/redaktion/Wahlen/handreichungen_gi_onlinewahlen.pdf*, 2007.

[12] R. Grimm and M. Volkamer. Multiple Cast in Online Voting - Analyzing Chances. *In R. Krimmer, editor, Electronic Voting 2006 - 2nd International Conference,* volume 86 of LNI, pages 97–106, Bonn, 2006.

[13] J. Heather, P. Y. A. Ryan, and V. Teague. Pretty good democracy for more expressive voting schemes. *In ESORICS*, pages 405–423, 2010.

[14] J. Helbach and J. Schwenk. Secure Internet Voting with Code Sheets. *In VOTE-ID*, pages 166–177. Springer, 2007.

[15] M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. *19th international conference on Theory and application of cryptographic techniques*, pages 539–556, 2000.

[16] T. Matsumoto. Human-computer cryptography: An attempt. *In Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 68–75. ACM, 1996.

[17] mIDentity from KOBIL. *http://www.kobil.com/index.php?id=49&L=0*.

[18] H. Nore. Open source remote electronic voting in norway. *Presentation at the Council of Europe*, Strasbourg 16/11-2010, 2010.

[19] R. Oppliger. How to address the secure platform problem for remote internetvoting. *SIS*, 2:153–173.

[20] R. Oppliger, J. Schwenk, and J. Helbach. Protecting Code Voting Against Vote Selling. *In Sicherheit*, pages 193–204, 2008.

[21] R. Oppliger, J. Schwenk, and C. Löhr. Captcha-based code voting. *In R. Krimmer and R. Grimm, editors, Electronic Voting*, volume 131 of LNI, pages 223–222. GI, 2008.

[22] D. Otten. Mehr Demokratie durch Internetwahlen? *Presentation at Nixdorf Forum,* Paderborn, 2005.

[23] S. Popoveniuc and P. L. Vora. Remote ballot casting with captchas. *http://www.seas.gwu.edu/~poorvi/RemoteBallotCasting.pdf*, 2008.

[24] R.L. Rivest. Electronic voting. *In Proceedings of Financial Cryptography 01*, pages 243 –268. Springer, 2001.

[25] P. Ryan and S. Schneider. Prêt à Voter with Re-encryption Mixes. *In Proceedings of Computer Security, ESORICS 2006*, pages 313–326. Springer, 2006.

[26] P.Y.A. Ryan and V. Teague. Pretty good democracy. *In Proceedings of the 17th International Workshop on Security Protocols*, Cambridge, UK, 2009.

[27] B. Schneier. The solitaire encryption algorithm, *http://www.schneier.com/solitaire.html*, 1999.

[28] G. Schryen and E. Rich. Security in Large-Scale Internet Elections: A Retrospective An alysis of Elections in Estonia, the Netherlands, and Switzerland. *Trans. Info. For. Sec.,* 4:729–744, December 2009.

[29] Trusted Computing Group. *http://www.trustedcomputinggroup.org/*.

[30] Team of oeh-wahl.gv.at. Sicherheitsempfehlung für Endbenutzer. *Technical report, http://www.oeh-wahl.gv.at/Content.Node/E-Voting_Sicherheitsempfehlung.pdf*, 2009.

[31] L. van Doorn. Trusted computing challenges. *In Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing*, *STC '07*, pages 1–1, New York, NY, USA, 2007. ACM.

[32] M. Volkamer. Evaluation of Electronic Voting - Requirements and Evaluation Procedures to Support Responsible Election Authorities, *Volume 30 of LNBIP*. Springer, 2009.

[33] M. Volkamer, A. Alkassar, A.R. Sadeghi, and S. Schulz. Enabling the Application of Open Systems Like PCs for Online Voting. *In Proceedings of Workshop on Frontiers in Electronic Elections*. Citeseer, 2006.

[34] T. Weigold, T. Kramp, R. Hermann, F. Höring, P. Buhler, and M. Baentsch. The Zurich Trusted Information Channel – An Efficient Defence Against Man-in-the-middle and Malicious Software Attacks. *Trusted Computing-Challenges and Applications*, pages 75–91, 2008.