

# An Intrusion-tolerant e-Voting Client System

André Zúquete  
IEETA / UA  
Campus Univ. de Santiago  
Aveiro, Portugal  
avz@det.ua.pt

Carlos Costa  
IEETA / UA  
Campus Univ. de Santiago  
Aveiro, Portugal  
ccosta@det.ua.pt

Miguel Romão  
UA  
Campus Univ. de Santiago  
Aveiro, Portugal  
a28244@alunos.det.ua.pt

## ABSTRACT

The ambition of any e-voting system is to reproduce, in an electronic environment, the characteristics of physical voting systems, such as accuracy, democracy, privacy and verifiability. REVS is an Internet e-voting system based on blind signatures and designed to be robust in distributed and faulty environments. However, the execution of REVS client system, used by voters, can be tampered by intruders willing to compromise the accuracy of submitted votes or the privacy of voters. In this document we present a new, intrusion tolerant e-voting client architecture for REVS. This architecture is based on public key cryptography, smart cards and FINREAD terminal readers.

## 1. INTRODUCTION

Internet voting systems are appealing for several reasons, one of them being the mobility of voters. Paper-based voting systems usually require voters to go to a specific voting poll. But using the Internet, voters may contact the right electoral servers virtually from anywhere in the world. However, the hosts used by voters to express their will must be trusted. Namely, they should not steal authentication credentials of voters nor interfere destructively with the voting process.

Trusted voting clients are difficult to implement in most hosts running general-purpose operating systems, such as Windows or Linux. The complexity of these systems and the degree of freedom in their configuration makes it nearly impossible to assure the correct behavior of a local voting client application. Therefore, critical parts of client voting applications should be deployed in restricted computing environments, capable of providing a proper Trusted Computing Base (TCB).

In this paper we describe an implementation of a voting client using a TCB composed by a FINREAD terminal and a smart card. The FINREAD terminal provides protection against disclosure for user input – authentication input, voter's choices – and output – presentation of ballots and voter's choices. The smart card provides pro-

tection for voter's authentication credentials – asymmetric key pair and voter's signatures – and the authentication of the received electoral data – ballot and information about electoral servers, all signed by an electoral authority.

By using this TCB we hope to be able to build an intrusion-tolerant, client voting system formed by the TCB and an ordinary personal computer (PC). The PC makes the required bridge between the TCB and the Internet. A compromised PC should, at most, interfere with the voting protocol using Denial of Service attacks.

For implementing and testing our voting client using this TCB we used REVS (Robust Electronic Voting System [10]). REVS is a publicly available Internet voting system where client voting applications are Java applets that run on voter's hosts. While our main goal was to change only the client application, in order to use the TCB when necessary, we had also to change other components of the REVS system – part of the voting protocol and some electoral servers – to introduce the authentication of voters with asymmetric key pairs.

This paper is organized as follows. Section 2 briefly describes the architecture of REVS, its voting protocol and the weaknesses of the client system. Section 3 describes the new REVS TCB-based client architecture. Section 4 describes some implementation details, regarding both modifications in REVS and the implementation of the TCB. Finally, Section 5 concludes the paper.

## 2. REVS VOTING PROTOCOL

REVS is a blind-signature based voting system designed for providing secure and robust electronic voting using the Internet [10]. The REVS architecture, depicted in Fig. 1, includes a client application (Voter Module), an electoral Commissioner, and a set of replicated electoral servers – Ballot Distributors, Administrators, Anonymizers and Counters. All these servers can be arbitrarily replicated for improving load balance, availability or for preventing collusion-based frauds.

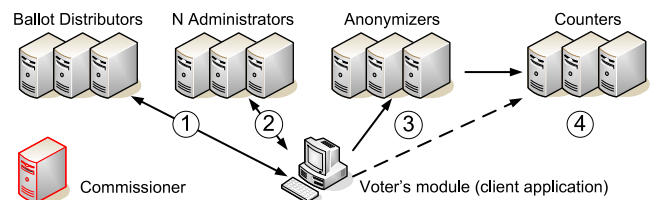


Figure 1: Architecture of REVS

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WRAITS 2007 Lisboa, Portugal

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

Fig. 1 also presents REVS protocol steps, which are:

**1 – Ballot distribution.** The Voter Module contacts a Ballot Distributor to get a ballot for a given election. The Ballot Distributor returns him the requested ballot, the election's public key and the election's operational configuration (e.g. servers' keys and locations); all this data was previously signed by the Commissioner. Ballots are XML documents providing a set of rules for presenting and verifying voting options for voters.

**2 – Vote signing.** After getting the voter's will for the ballot, the Voter Module commits to the vote with a random bit string (bit commitment) and generates a digest of the committed vote. Then the Voter's Module generates random blinding factors, applies them to the digest and sends the results to a subset of the  $N$  Administrators for signing. Messages sent to Administrators are authenticated by voters using a username/password pair, in order to ensure that only authorized voters participate in the election. An Administrator, after receiving an authenticated, signing request, verifies if it had already signed a blind digest for the requesting voter. If not, it signs the vote and saves that signature; if it had signed before, it returns the previously saved data. After receiving the signature, the Voter's Module removes the blinding and verifies its correctness using the Administrator's public key. This process is repeated until a required number of  $t$  signatures is collected. The value of  $t$  must be higher than  $N/2$ , to prevent voters from getting more than one valid vote.

Each voter uses a different password for each of the  $N$  Administrators. This is automatically managed by the Voter's Module: the voter introduces only a PIN and a password and the Voter Module uses a cyclic, digest-based password generation process to generate the password for each Administrator. To prevent eavesdropping, the passwords are sent directly to Administrators using SSL secure channels with server-side authentication.

**3 – Vote submission.** The Voter's Module constructs the vote submission package, joining the vote, its signatures and the bit commitment and encrypting everything with the election's public key. Then he submits this package to any number of the Counters through the Anonymizers, concluding the voting protocol.

**4 – Vote tallying.** At the end of election Counters merge and publish all submitted packages. The Commissioner discloses the election private key and anyone is able perform the tally: decrypt the submitted packages, discard replicated votes (the ones with the same bit commitment), validate signatures on remaining votes and perform the tally with the valid votes. Missing packages can be detected and resubmitted anonymously by voters.

## 2.1 Weaknesses in the Voter Module

For portability, the Voter Module is an ordinary Java application. The Voter System, formed by this application and the system where it runs – Java Virtual Machine (JVM) and host system – may be tampered in many ways in order to interfere with the privacy of local voters or with the accuracy of an election. Possible malicious actions taken by the Voter System are the following:

- Provide the voter with false ballots or sending false votes instead of the ones filled by the voter (accuracy issue).

- Steel voter's credentials – identification and authentication passwords (impersonation issue).
- Use otherwise the identification of the voter and his votes (anonymity issue).

To improve the privacy of the voter and accuracy of its participation in the election, the Voter System should use a TCB for:

- Protecting the credentials – the public key of the Commissioner – used to validate the data received from Ballot Distributors.
- Protecting the input of voter's choices for ballots.
- Protecting the voter identity and authentication credentials from steeling.
- Protecting all sensitive information related with the voting protocol – bit commitment and blinding factors.

This approach requires moving part of the Voter Module into a TCB, in order to protect critical validation data (e.g., the Commissioner's public key), voter's critical input/output operations and all permanent or temporary, personal data involved in the voting process.

## 3. NEW TCB-BASED VOTING CLIENT SYSTEM

The present contribution was developed to improve the REVS e-voting system, namely its Voter System. As we saw before, this component raises some critical accuracy, impersonation and privacy issues when relying on ordinary, insecure hosts. Therefore, it can become the Achilles heel of the whole REVS system.

Our main goal was to develop an intrusion-tolerant voting client system by migrating part of the REVS Voter Module into a TCB, providing the required protection of private data and preventing vote tampering. For implementing the TCB we chose a trustable smart card/smart-terminal environment, namely a smart card and a FINREAD device. These two components are described in more detail in the two following sections.

Voting clients using smart cards are not new, though not frequent [3, 4, 9]. They use smart cards mainly for making computations with voters' private keys in a personal, secure computing environment. We go one step further, by using the smart card for storing other sensitive data, such as the Commissioner's public key and cryptographic values used for creating the vote. Furthermore, we protect the interaction between the voter and the smart card by means of a secure I/O terminal with enough human-interface capabilities – a FINREAD reader. To the best of our knowledge, this is the first time a FINREAD reader is used for supporting secure voting clients.

### 3.1 Smart Cards

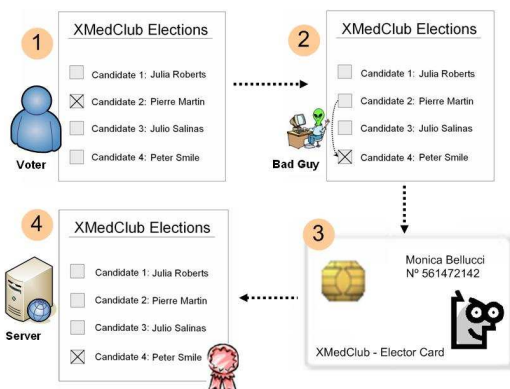
Smart cards provide a properly protected support for storing sensitive, private information, such as personal details or cryptographic keys [11]. There are various ways to use this technology [7], but when correctly combined with other security technologies, such as public key cryptography (PKC) and biometrics, it strongly enforces effective access control through personal identification and/or authentication [8].

There are various types of smart cards, but the most interesting in terms of security for implementing our TCB are those that have an embedded microprocessor capable of executing strong cryptographic algorithms on the card itself, thus not requiring protected information to be moved from the card. The use of those storage and processing devices, with native cryptographic capabilities and protected by user-provided secrets, can improve the level of security to the whole system. Smart cards are an ideal solution for PKC authentication: the private key lies in a secure, tamper resistance storage, a second factor authentication (a PIN) must be introduced to unlock it and a crypto accelerator provides cryptographic hardware operations, such as asymmetric key pair generation and digital signature generation/verification. We have been working with two distinct types of smart cards proving these functionalities: Schlumberger (Axalto) Cryptoflex (16 kB) and Javacard Cyberflex egate (32 kB).

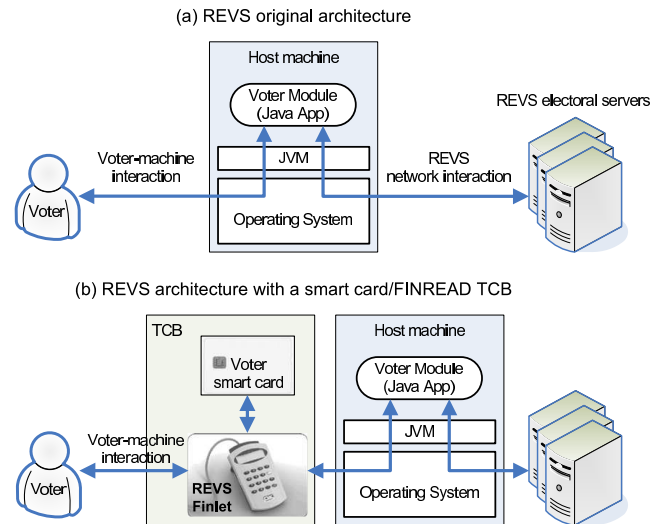
PKC-based authentication systems are more secure than password-based systems because there is no shared knowledge of secrets between transaction intervenients. The private key, used to compute an authentication signature, needs only to be known by one side of transaction – the one being authenticated. In a typical PKC authentication process, the entity being authenticated signs some data with its own private key and the authenticator validates the data and the signature using the corresponding public key.

Considering REVS, a voter proves his authenticity to Administrators by signing requests with a PIN protected, private key stored inside a smart card. Administrators verify the authenticity of a voter by checking his signatures with his public key stored in their databases. This public key must be fetched from the voter's smart card during his registration phase.

By providing a voter with a PKC digital credential, stored and protected by a smart card, we can enforce strong voter authentication and protected, authentication signatures performed inside the smart card. However, smart cards cannot evaluate the correctness of data provided for signing. Therefore, they cannot prevent the adulteration of votes by a tampered Voter System running on a PC environment (see Fig. 2); this issue is handled by using a trusted card reader.



**Figure 2: Vote adulteration by the Voter Module:** (1) the voter fills the ballot; (2) the Voter Module tampers the data sent for smart card signing; (3) the smart card signs the provided data; and (4) a tampered vote is submitted



**Figure 3: Evolution in the architecture of REVS for support a smart card/FINREAD TCB in the voter side**

### 3.2 OMNIKEY CardMan Trust FINREAD

As previously explained, the usage of a smart card does not prevent the possibility of vote adulteration in the client PC environment. For instance, REVS Java-based client module could be cracked or maliciously cloned without being detected by the voter. Since steps 1 and 2 of Fig. 2 are actually executed in the PC environment, this represents a potential risk to the voting process.

After analyzing some possibilities to enforce a secure human-machine I/O in the client voting environment, we decided to separate this component from the Voter Module running on the PC, using instead a secure I/O device. The evaluation of several tamperproof devices conducted us to elect the FINREAD smart card terminal as a promising option.

Therefore, our TCB was built around a Java-based smart card reader, namely the Omnikey CardMan Trust FINREAD [12]. This is an ISO 7816 and EMV 3.1.1 compliant smart card reader [1], which also adopts the latest FINREAD specifications [5, 2].

The FINREAD platform adopts and extends the Java applet technology; in a FINREAD environment applets are called Finlets. FINREAD terminals are provided with internal memory (1MB) capable of hosting different software modules (Finlets) and a JVM to execute them. When a Finlet is activated in the terminal, it starts operating in secure mode, i.e. the access to the smart card is always mediated by a Finlet [12]. Otherwise, it operates in transparent mode, acting as an ordinary smart card reader.

Our FINREAD device has reduced but enough human-machine I/O capabilities. It has a small LCD display, containing 4 lines of 20 characters, and a 16 keys pin-pad for user input interaction. Finlets running on the terminal can control both sensitive transactions with the smart card and interactions with the user (voter) using the terminal's display and pin-pad. Namely, the vote options can be displayed by the terminal, the voter choices are introduced through the pin-pad, as well as the smart card PIN.

TCB	Functionalities	
Smart card	Storage	Commissioner public key Voter's authentication asymmetric key pair Per-vote temporary data (vote, bit commitment and blinding factors)
	Computing	Generation of random data (bit commitment and blinding factors) Generation of voter's authentication signatures Digest computations
FINREAD terminal	I/O	Vote presentation and input Smart card PIN input
	Storage	Per-election public keys (Administrators, election) Per-election Administrators' signatures
	Computing	Validation of data provided by Ballot Distributors (signed by the Commissioner) Blinding and unblinding of data exchanged with Administrators Validation of Administrator's blind signatures Production of the vote submission package

Table 1: Functionality provided by the smart card and FINREAD terminal

### 3.3 TCB Functionality

We considered that this set of components, a FINREAD terminal and a smart card, was suitable for implementing a TCB assuring the required level of security for the new REVS Voter System. The new architecture of REVS using this TCB is presented in Fig. 3. The TCB components provide for REVS the set of functionalities presented in Table 1.

The smart card is used to store sensitive data, private data or critical data for resuming the voting process latter if a fault occurs. The Commissioner's public key is sensitive data, as it is used to validate all the data received by the voter from electoral servers. The voter's asymmetric key pair, his vote, the vote's bit commitment and all blinding factors used in blind signature procedures are both private and critical. Note that this critical data must be stored in non-volatile memory, such as a smart card, to resume the voting process latter if a fault occurs. Storing this data in the smart card also allows the voter to use another FINREAD terminal to complete his voting process.

On the contrary, the FINREAD only stores temporary data that is required to perform local computations – the public keys of electoral servers and valid signatures provided by Administrators. This data can always be recovered after a fault during the voting process.

### 3.4 Changes in REVS protocol

The REVS voting protocol was only modified to support the PKC-based authentication of voters. In the original protocol each request sent to an Administrator was formed by an election ID, a voter ID and password and a blinded hash of the vote and bit commitment:

$$\text{old request} = \text{election ID, voter ID, password,} \\ \text{blinded}(\text{hash}(\text{vote, bit commitment}))$$

The password for each Administrator was presented by the voter in the registration phase and distributed by the Commissioner to the proper Administrator.

In the new protocol, a voter gets authenticated using an asymmetric key pair produced by his smart card. This key pair can be produced during the registration phase, if not already present in the card, and the Commissioner fetches the voter public key from the card and sends it to all Administrators. During the voting process, each request sent to an Administrator is now formed by an election ID, a voter ID, a blind hash of the vote and bit commitment and a voter signature of all this data. For preventing eavesdropping by

a malicious Voter Module, both election ID and voter ID padded with a random value and encrypted with the Administrator's public key:

$$\text{new request} = \{\text{election ID, voter ID, random}\}_{\text{Adm } K_{\text{pub}}}, \\ \text{blinded}(\text{hash}(\text{vote, bit commitment})), \\ \text{voter signature}$$

### 3.5 FINREAD human interface issues

Because the REVS platform is fully implemented in Java, at the beginning it may appear that it would be relatively easy to migrate selected parts of the original Voter Module into one or more FINREAD Finlets. However, this migration is more complex due to natural limitations of the FINREAD terminal. Other limitations, presented by the FINREAD JVM, will be presented in Section 4.1.

As we have already highlighted, the FINREAD device not only allowed us to prevent the PIN capture outside the smart card reader, but also to display the ballot and retrieve the voter choices. Like all blind signature based voting systems [6], REVS is ballot independent. Ballots are XML documents, which can be easily transferred to the terminal and parsed inside it for proper presentation and fulfillment.

Due to the FINREAD display limitations, the presentation of the ballot in this device must be completely changed. Furthermore, the ballots' text to present to voters must be produced differently, namely using reduced amounts of data and tacking into consideration the reduced capabilities of a FINREAD display. A complementary approach is to produce in the FINREAD terminal, from the same ballot, a rich, graphical image for presenting in the PC display. This image could facilitate the understanding of the ballot options by the voter. But for the privacy of the voter, it must never present the voter choices and the graphical image must not be easily tampered by the PC, something that may be difficult to accomplish.

## 4. IMPLEMENTATION

The implementation of this TCB in REVS was done in two phases. In the first phase we changed the protocol to use PKC-based authentication of voters. Namely, we (i) changed the Commissioner to register voters' public keys fetched from smart cards and to store its own public key in the smart cards, (ii) changed the Administrators to use voters' public keys for authentication and (iii) changed the

Voter Module to use a smart card to sign requests sent to Administrators. During this phase we have used ordinary smart card readers, or the FINREAD reader in transparent mode [12], and the IAIK Java Cryptographic Extension [13].

In a second phase we developed a Finlet for implementing parts of the Voter Module within the FINREAD terminal. Currently the Finlet functionality includes 3 components.

The first component receives the data retrieved from a Ballot Distributor and validates it with the Commissioner public key stored in the smart card. The vote is interpreted, presented and filled using the FINREAD display and pin-pad. A random bit commitment and several random blinding factors are generated using the smart card random generator. At this stage the vote and all the previous random values are stored in the card for fault tolerance. The smart card PIN is read using the FINREAD pin-pad and the requests for the all the Administrators are produced and signed using the smart card and the voter private key. At the end, this component returns to the Voter Module all the requests that must be sent to Administrators.

The second component receives a signature from an Administrator and validates it with the Administrator's public key received by the first component and stored in the FINREAD memory. If the signature is correct it is stored in the FINREAD memory for latter use.

The third component packs the vote, the bit commitment and the required number of valid signatures collected from the Administrators and builds the submission package. This package is then returned to the Voter Module for being submitted to Counters through Anonymizers.

The Voter Module uses the first component after getting data from a Ballot Distributor, uses the second component after getting each signature from an Administrator and uses the third component after getting  $t > N/2$  valid signatures from Administrators.

Along all this process, the Voter Module that runs in the PC has no direct influence in the presentation and ballot filling out, no access to the voter's authentication credentials – smart card PIN and private key – and no direct access to the vote – the vote submission package can only be open at the end of the election. Furthermore, the identification of voters sent to Administrators, when getting their authorization signature, is encrypted by the terminal with the Administrators' public keys.

## 4.1 Implementation issues

The FINREAD execution environment is quite unique. Finlets are event-driven applications but there is no event loop; methods with standard names are called for handling events. There are no debugging facilities other than sending messages to the FINREAD display or to the PC.

The FINREAD JVM misses some useful functionalities that could facilitate our work: there is no support for XML parsing and for serialization. The lack of XML parsing support forced us to implement a small, simple parser within the Finlet for handling ballots. The lack of serialization support complicated the interface between Voter Module and Finlet and forced us to build differently vote submission packages and to deploy a new interface in Counters.

## 5. CONCLUSIONS

There are a great number of sensitive services available on the Internet and this number is expected to continue to grow

in the next few years. One of these services is electronic voting, that should accomplish the desired characteristics of traditional voting systems, such as accuracy, democracy, privacy and verifiability.

Our working platform was REVS, a robust electronic voting system designed for distributed and faulty environments, namely the Internet. The Voter Module of REVS was designed for trusted hosts, thus raising privacy and accuracy concerns when considering its widespread use in any host. To handle this problem we designed a TCB, using a smart card and a card reader with human-machine interface capabilities, to enforce the trusted execution of critical parts of the Voter Module in untrusted or compromised hosts.

In this paper we described a new architecture for the REVS Voter Module using this TCB. The result is a voting client system that is intrusion tolerant. We also modified the authentication process of voters in REVS – from user-name/password pair into asymmetric key pairs – to benefit from the use of PKC-based authentication provided by the smart card. The final system provides a robust PKC-based authentication of voters and protects all critical actions and data of voters, during the voting process, with tamper-proof devices – smart card and FINREAD reader.

## 6. REFERENCES

- [1] Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4 : Interindustry commands for interchange. ISO/IEC 7816-4, 1995.
- [2] OMNIKEY FINREAD SDK Manual, v1.22.3, 2005.
- [3] C.-B. Breunese, B. Jacobs, and M. Oostdijk. Voting using Java Card smart cards: A case study, Jan. 2002.
- [4] S. Canard and H. Siberty. How to fit cryptographic e-voting into smart cards. In *Frontiers in Electronic Elections (FEE 2006)*, Hamburg, Germany, Sept. 2006.
- [5] F. Consortium. FINREAD Technical Specifications, Parts 1-8, 2003.
- [6] A. Fujioka, T. Okamoto, and K. Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Adv. in Cryptology – AUSCRYPT '92 Proc. (LNCS 718)*, Queensland, Australia, 1992. Springer-Verlag.
- [7] H. Gobioff, S. Smith, J. D. Tygar, and B. Yee. Smart Cards in Hostile Environments. In *2nd USENIX Works. on Electronic Commerce*, Oakland, USA, 1996.
- [8] G. Hachez, F. Koeune, and J. Quisquater. Biometrics, Access Control, Smart Cards: a Not So Simple Combination. *Security Focus Magazine*, Oct. 2001.
- [9] J.-K. Jan and C.-C. Tai. A Secure Electronic Voting Protocol with IC Cards. *Journal of Systems and Software*, 39(2), Nov. 1997.
- [10] R. Joaquim, A. Zúquete, and P. Ferreira. REVS – A Robust Electronic Voting System. *IADIS Int. Journal of WWW/Internet*, 1(2), Dec. 2003.
- [11] R. Marvie, M.-C. Pellegrini, O. Potonniée, and S. Jean. Value-added Services: How to Benefit from Smart Cards. In *Gemplus Developer Conf. (GDC 2000)*, Montpellier, France, June 2000.
- [12] K. Schmid and H. Zeitlhofer. FINREAD Whitepaper, Rev 1.0, 2003.
- [13] I. SIC. The IAIK Provider for the Java Cryptography Extension (IAIK-JCE), 2001.